ISY 2014
Intelligent Systems Workshop

July 10th 2014
Bielefeld University, Bielefeld, Germany
http://techfak.uni-bielefeld.de/isy-praktikum/

Proceedings

Edited by Thomas Hermann, Kevin Fischer and Florian Lier

CONTENTS

INTELLIGENT SYSTEMS PROJECT: AUGMENTED INTELLIGENT SPACE

B. Errouane, J. Platte, K. Klein, R. Schiewer

Faculty of Technology, Bielefeld University
Bielefeld, Germany

Supervisor: T. Pfeiffer

ABSTRACT

The goal of this project was to provide a system, with which the user can navigate through and interact with a virtual space that represents the real-world environment. To ensure the immersion we use a head-mounted display (HMD) that keeps track of the user's head rotation and a marker tracking software that determines the user's head movement utilizing a camera attached to the HMD. With movable markers and a flexible two-handed input device we allow the user to interact with the virtual world that is being simulated in the Blender Game Engine. Using a flexible middleware we created a modular software system that is easily expandable.

1. INTRODUCTION

When dealing for example with problems in a cognitive information technology context, there are often lots of sensors, actuators and 3rd party applications involved, all connected via a middleware. Hence keeping track of things can become a difficult task. To carry out simple tasks like e.g. verify the output of a sensor or, in general, a system component, the user often has to do the same steps, (addressing components, query data etc.) over and over again. In contrast to [1], where the focus was to provide an agent-based intelligent system, to facilitate user tasks, we use a rather simple and more controllable approach. We considered using a complex intelligent system would be a by far to extreme strategy for what we want to achieve.

The idea behind AIS is to provide additional interfaces to simplify and speed up the process of interaction by providing firsthand information inside the virtual representation of an environment. We thereby use a similar approach like [3], but with a more lightweight software architecture and a non-see-through display. Due to the simplified interaction principle, this approach also allows untrained users to work with complex systems. Furthermore, in contrast to [3] we do not concentrate on the human-robot-interaction in particular and do not have to deal with the problems, that accompany an augmented reality approach, because we use the non-see-through display.
The main goals of AIS are as follows:

- Display a virtual representation of the users environment to the user via a head-mounted display.
- Provide an intuitive control mechanism to manipulate the viewport of the rendered scene.
- Integrate several interaction interfaces for the user to select and manipulate in the virtual space.

2. SYSTEM DESIGN

The system we developed consists of several single components. We use a head mounted display to visualize the scene, one or more cameras for marker tracking purposes, a Razer Hydra for advanced user inputs and a Microsoft Kinect camera for displaying real world content within the simulation.

To display the virtual scenery to the user we utilize the Oculus Rift head-mounted display. We have chosen the Oculus Rift, because on the one hand it is a well known device and widely spread, thus there is already much support and knowledge available. On the other hand, we wanted a non-see-through HMD (head-mounted display), because that way we maintain the full control over what the user can see and what not. A third argument to favor the Oculus Rift is the integrated accelerometer, which allows us to track the user's head turns in real time. The positional tracking of the user's head was not possible with the early prototype of the HMD that we used. Unfortunately, the resolution of the Oculus Rift's display is only 640x400 pixels per eye, which results in a very coarse grained image. But considering the fact, that the next version of the Rift was already in development by the start of our project, the low resolution was of minor importance.

To track the user's position inside the test environment, we use a simple marker tracking approach, whereas for marker detection we use a common webcam. Our first approach was to stick a marker the to HMD and track the position of this marker using a fixed camera with a known position in the room. This worked fine but the space that the user was tracked in was very limited. To extend this space we would have to use multiple cameras all over the room. We then changed our scenario to attached the camera itself to the HMD via a custom made 3D printed mount. We printed out several markers and placed them in the room in spots for the camera to see them and thus an algorithm we developed (further explained in section 3.1) can determine the user's position in the environment. The markers are ordinary VR markers and theoretically replaceable, if another marker-system could be found as good or better. Therefore, the marker tracking library would have to be replaced as well. The advantage of this solution is that we only have to use the information of one webcam. Also it is way easier to print out markers and place them in the room, than to do this with webcams that all have to be calibrated and connected with a pc.

The model of the environment was created in Blender. This allowed the use of the Blender Game Engine, which offers an easy to use simulation framework in which the user's actions can be processed and presented back to the user. Using the scripting mechanism of the Game Engine the other software components' data could be accessed easily.

Using a mouse and keyboard as input devices would bring the advantage that most users are used to these devices. But in our scenario with an HMD the usage of those is not very advisable as you can not see your hands and it would hinder your mobility as both devices need a solid ground to work properly. So as interaction device the Razer Hydra was chosen, because it offers 6 degrees of freedom additionally to four buttons, one trigger button and an analog stick for each hand. This allowed a wide variety of possibilities to interact with the system. Besides the data of the stick has low input lag and a high precision. The Razer Hydra has two position and orientation detecting controllers, which are connected by wires with a base, thus the 6 degrees of freedom. This base creates a weak magnetic field in which the controllers can be detected very accurately. Figure 1 shows a user with the controllers using our system.

We also use the Microsoft Kinect camera to display arbitrary real world content in the simulation. Thus, it serves as a link between real and simulated environment and is part of our interaction scenario. The Kinect records a RGB and a depth image, whereas both images have a resolution of 640x480 pixels.

To connect all of these components RSB (Robotics Service Bus) was chosen as it is easily deployable and allows to flexibly add more modules. This way it was possible to easily add the Microsoft Kinect in a late stage of the project. Further details about the interconnection between these modules are outlined in 2.1.

2.1. Interconnection

With RSB different processes can communicate via the Spread Toolkit. The bus provided by RSB is organized with hierarchical scopes to which processes can send messages and listen for messages of other processes.[1]

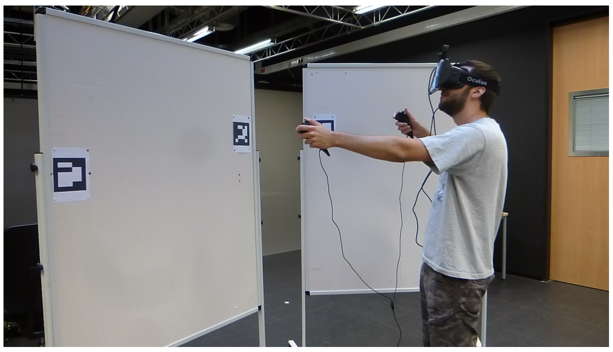


Figure 1: A user of the system. You can see the HMD with the mounted webcam and the two Razer Hydra controllers. In the background several markers are visible that are used to calculate the user's head position.

In this project we made heavy use of this functionality. For example the marker tracker sends all found markers to a certain scope for every camera frame and does not check whether the data is being received by the calibration process or not. The calibration application on the other hand listens to said scope and reacts if it receives found markers. The details of the calibration are described in section 3.1.

The same method was used to send the data of the different modules (Microsoft Kinect, Razer Hydra and Marker Tracker) to a receiver which can be read by the Blender Game Engine. In section 3.4 you can find further information about this receiver.

With RSB it would easily be possible to distribute the modules to different machines. With several sensors and processing steps this option would be preferable to distribute the workload because it would result in a better overall performance. But in our case this was not necessary.

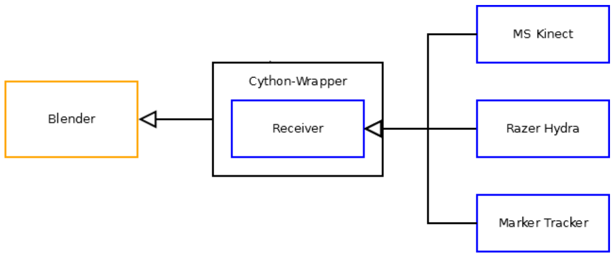


Figure 2: Connections between different modules. A blue frame denotes a component written in C++

[1] see https://code.cor-lab.org/projects/rsb

3. SOFTWARE COMPONENTS

3.1. Marker Tracker & Calibration

If a camera is calibrated using the technique described in [7], the used marker tracking software provides the transformation matrices of markers that are visible to the camera. These transformations are relative to the camera's coordinate system. The markers can be differentiated by an ID which is encoded in the pattern of the markers themselves. If the software saved these markers' positions in an earlier step, the camera's position in the marker's coordinate system can derived. There are two calibration steps necessary for this process to work.

The first step is the calibration of the camera by showing it a uniform chessboard printout. A calibration program uses this image information to calculate the intrinsic parameters of the camera and saves these to an XML-file. This file is later on used by the marker tracking software to calculate the markers' positions with respect to the camera.[7]

The second step is the calibration of the markers' positions in the room. As these markers are used to determine the user's position, they may not be moved later on. Using the intrinsic parameters previously gained, the marker-tracker sends the marker transformations with respect to the camera to the calibration software via RSB as explained in 2.1. The calibration software receives these markers and handles them as follows:

- If the found marker is the first one to be received, it is saved as the origin of the world coordinate system. It's transformation-matrix is a 4x4 identity matrix. The marker is further on called a *known* marker.
- If the found marker is an *unknown* marker, but a *known* marker was also found in the same frame, it's transformation in the world space is saved and the marker is further on *known* to the system. The world transformation $T_{u,w}$ can be calculated using the following formula:

  $$T_{u,w} = T_{u,c} * T_{k,c}^{-1} * T_{k,w}$$

  where the index u denotes the unknown and k the known marker. Transformations with index c are with respect to the camera coordinate system and the index w corresponds to matrices relative to the world coordinate system. Multiplication of transformation matrices is equal to the successive application of them[2]. So in this case the result is exactly the matrix we want. Figure 3 helps to get an idea what the transformations represent.
- In any other case, the marker is ignored

[2]see http://www.3dcoding.de/2011/10/order-of-matrix-multipliation-and-its-effect-on-transformations/

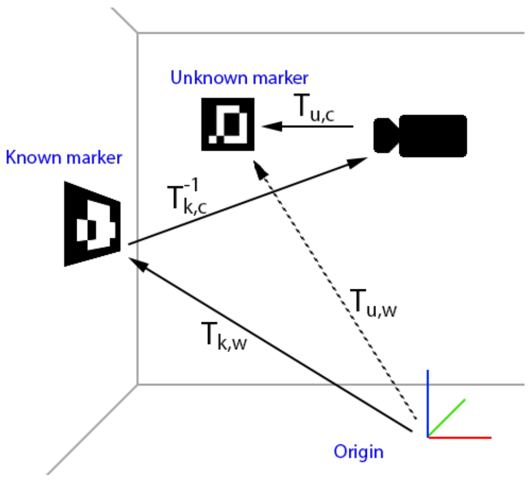


Figure 3: Visualization of the calibration process and the matrices involved (represented by arrows). Both markers are in the camera's image.

When the calibration is done, the marker-tracker receives the calibration data which includes the marker IDs and the corresponding transformation matrices in the world coordinate system.

From now on if any *known* marker is found in the camera image, the camera position in world coordinates is being sent to the Blender Game-Engine using RSB and the cython-wrapper explained in 3.4. The game engine changes the scene's camera position accordingly. This way the user's position in the virtual reality always corresponds to his real-world position if a known marker is visible to the camera.

Unknown markers world transformations are being sent to the game engine as well to be able to move virtual objects. These marker's transformations $T_{u,w}$ are calculated using the formula:

$$T_{u,w} = T_{u,c} * T_{camera}$$

where the indices are the same as before, but T_{camera} is the last known camera position.

This is used (instead of calculating the current position with a known marker's transformation), to provide a fluent visualization of marker movement to the user. Otherwise the marker movement of unknown markers would not be represented in the virtual reality if no known marker is visible and thus the immersion of the virtual reality would probably suffer.

3.2. Razer Hydra

While using the the Razer Hydra, a problem is that there is a difference between the coordinate system in which the Razer Hydra controllers are represented and the coordinate system of Blender. In both systems the x-axis points to the right. But the y-axis for the Razer Hydra points up while in Blender it points forward, into the screen, and the z-axis for the Razer Hydra points backwards, out of the screen, while it points up in Blender. This behavior is shown in figure 4.

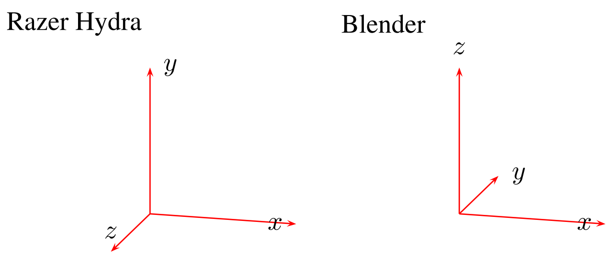


Figure 4: Different coordinate systems for the Razer Hydra and Blender

To use the data the Razer Hydra controllers provide us in Blender, we have to modify it. For the positions we only have to swap the y- and z-coordinate and after that change the sign of the new y-coordinate.

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \mapsto \begin{pmatrix} x \\ -z \\ y \end{pmatrix}$$

Note that the data received from the Razer Hydra is given in millimeters, while we use one Blender unit as one meter, so we have to divide the position data by a factor of 1000.

A bit more difficult is the mapping for the rotation matrix. As the columns in the rotation matrix are orthogonal basis vectors, we have to do the following[3]. The first step is to swap the rows like we did above.

$$\begin{pmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ z_1 & z_2 & z_3 \end{pmatrix} \mapsto \begin{pmatrix} x_1 & x_2 & x_3 \\ -z_1 & -z_2 & -z_3 \\ y_1 & y_2 & y_3 \end{pmatrix}$$

Then we operate the same mapping to the columns of the matrix.

$$\begin{pmatrix} x_1 & x_2 & x_3 \\ -z_1 & -z_2 & -z_3 \\ y_1 & y_2 & y_3 \end{pmatrix} \mapsto \begin{pmatrix} x_1 & -x_3 & x_2 \\ -z_1 & z_3 & -z_2 \\ y_1 & -y_3 & y_2 \end{pmatrix}$$

So all in all we come up with the following mapping for the rotation matrices:

$$\begin{pmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ z_1 & z_2 & z_3 \end{pmatrix} \mapsto \begin{pmatrix} x_1 & -x_3 & x_2 \\ -z_1 & z_3 & -z_2 \\ y_1 & -y_3 & y_2 \end{pmatrix}$$

[3]see http://sixense.com/forum/vbulletin/showthread.php?3591-Tutorial-2-Coordinate-Space

3.3. Interaction

Sticking to these mappings we can use the data of the Razer Hydra in Blender to rotate and move a model of two hands and perform interactions.

To interact with something, e.g. a camera, we create a semitransparent red ray which has the left hand's position as its origin and the difference of the right hand's position and the left hand's position as the direction.

For the rotation of this ray we use the Rodrigues matrix which rotates by any given angle about a given axis[4]. We model the ray pointing up, so by using the cross product of $(0, 0, 1)^T$ and the normalized direction vector between the hands, we get the rotation axis and the sine of the angle. For the specific rotation matrix we also need the cosine which is given by the dot product of the same two vectors.

For the manipulation of a camera, we damped the rotation to achieve a more accurate handling of the rotation. Euler angles are used for this damping. As a damping factor we decided to take $\frac{1}{5}$ for each Euler angle which gives the user a quite good way to rotate the camera as much as intended.

To avoid toggling of the rotation due to the damping after rotating about more than 180 degree, we fix the old rotation if the norm of the difference of the previous and the current Euler angles is greater than a certain threshold.

3.4. Cython Wrapper

As already stated, most of the external data is transported to and from Blender via RSB. The gap between the arriving or departing data on the host computer (the one, which runs Blender) and the use of this data in Blender is filled with a custom written receiver class.

We use Cython to wrap a C++ class within a python-library for use within a python application and thereby effectively connect Blender and RSB[5]. The Cython wrapper offers functionality to query different sensors, that are expected to send data via RSB. The query-and-use pattern is always the same:

1. Within Blender script: Poll data cache (Cython wrapper function) regularly
2. If data was obtained (check e.g. length of received buffer), use it

One advantage of this strategy is, that if data was received, it stays in the buffer until the application is shut down or new data arrives. Thus, the receiving and handling of external data is properly separated and one can even turn off a sensor (e.g. for debugging purposes) and continue work with the remaining data in the buffer. Furthermore,

[4]see http://mathworld.wolfram.com/RodriguesRotationFormula.html
[5]see http://docs.cython.org/src/userguide/wrapping_CPlusPlus.html

adding new sensors was not very time-consuming, since the design steps are always the same. Due to this clear definition of our interfaces, we even could develop the server application, which queries the sensor and publishes data onto a RSB scope, and the data processing script in Blender at the same time. All that was left to do was, to extend the functionality of the receiver class, which involved always the same steps.

3.5. Oculus Rift Components

The connection of the Oculus Rift to our application illustrates an exception in our design pattern, because the Rift hast do be connected to the host computer directly. There is no detour via RSB, whereas we use Cython again to wrap needed functions of the openHMD driver and thus obtain necessary data. This exception primary exists, because the Rift was the first external sensor, we added to our application, and since we did not have any experience regarding the lag of data transfer caused by RSB, we decided to connect it directly.

The as fast as possible processing of the user's head turns on the one hand and the depiction of the rendered scene on the Rift's display on the other hand seemed to be very time-critical to us, so delaying one of them or both without a real reason seemed not feasible.

To view the scene on the HMD correctly, it has to be distorted on software side to compensate the distortion of the Oculus' lenses. To do so we basically programmed a shader like the solution proposed in the Oculus Rift developers manual[6] and call it during every tick of the Blender Game Engine.

3.6. Microsoft Kinect

Basically to give an example for a device, which can provide a useful extension for our system, we depict the depth image of a Kinect camera within our simulation. The incoming data buffer of the Kinect is read and then sampled to create an image $1/10$th as precise as the original depth image, which we do because of performance issues.

In a next step, the pixel data of the $64 \cdot 48$ pixels containing image is element by element transferred to a 2D plane, which consists of exactly the same number of vertices. Whereas we first modified the relative z-component of the vertices and thus displayed a coarse 3D interpretation of the depth image, we then decided to simply show the (2D) depth image itself.

We did this, because the 3D interpretation was not recognizable (even in higher resolutions) and the modification of the vertex positions was computationally out of scale. To ensure, that every vertex gets the corresponding pixel's color, the correct ordering of the 2D plane's vertices and the information in the depth image's buffer is vital.

The buffer is received as a 1D row major array of unsigned chars from lower left to upper right corner, so the vertices have to be ordered row by row from left to right and bottom to top. Because we could not find any system in the vertex order of the planes that were created with Blender, we therefor now sort the plane's vertices at every program start and save a list, which holds the proper sequence of vertex indices. The correct position in this list for a vertex v is determined trough a 1D coordinate c_v, which is computed as follows:

$$c_v = p_y \cdot 10 + p_x,$$

where p_x is the vertex' x-coordinate and p_y is it's y-coordinate. All computed c_v are saved together with the corresponding old vertex index $i_{v,old}$. Finally, when all c_v are known, then list of $< c_v, i_{v,old} >$ tuples is reordered with respect to c_v and the second parts of all tuples, namely the $i_{v,old}$ are saved in another list. This sequence then gives the correct order, in which the vertices have to be processed.

3.7. Model of the World

To be able to move trough a real, existing environment with the Oculus Rift, a virtual model of this environment is needed. As we conducted most of our research work in the Virtual Reality Lab of the CITEC, we decided to create a model of this laboratory.

The model is made in Blender and features high resolution textures and detailed structures, as well as furniture. Also, different entities can be enabled or disabled through the layer mechanism of Blender.

The motivation behind creating a complex and costly scene like this is, that one of the main parameters to achieve good immersion is, to show a plausible and realistic looking world to the user. So we did not want our approach to fail just because of a qualitatively poor made 3D model.

4. INTERACTION/OPERATION EXAMPLES

As an interaction scenario we thought of a task in which someone has to arrange several cameras to cover a wide area, e.g. in the so-called CAVE (Cave Automatic Virtual Environment).

With our system we can model the cameras with their viewing frustum, so you can see quickly which area is not yet covered and where the frustums of multiple cameras are overlapping. Then you can rotate the cameras the way most of the area is covered, print out these rotations and apply them to the original cameras.

After calibrating the camera and the marker setup, as described in section 3.1, and starting the Kinect and the

[6]see http://static.oculusvr.com/sdk-downloads/documents/Oculus_SDK_Overview.pdf, last visited: 29.06.2014

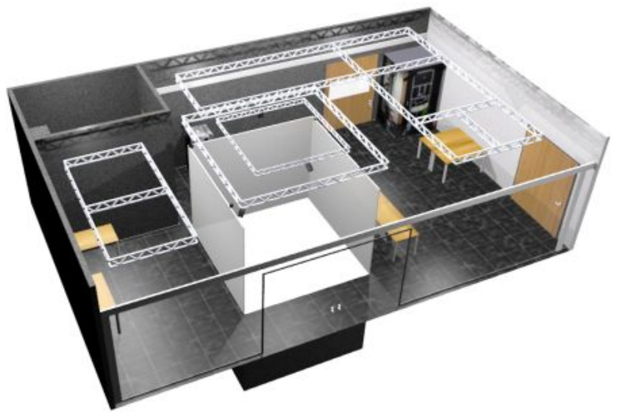


Figure 5: Blender-Model of the CAVE-Laboratory

Razer Hydra components, note that the Razer Hydra controllers have to be placed onto the base for initialization, the Blender Game Engine can be started and the user can put on the Oculus Rift.

Now the user is able to move around and while seeing at least one marker, the position of the user in the Blender model changes correspondingly just as the orientation of the Oculus Rift changes the orientation of the user in Blender. Thus an immersive feeling should be created.

Next the Razer Hydra can be used to perform some interaction with the virtual space. Moving the two controllers moves the model of two hands accordingly and thereby the ray cast by these hands. If a camera is on the ray, the user can push the trigger button of the Razer Hydra controller which is closer to the camera to select the camera for manipulation.

After that the ray disappears and by rotating this controller, the camera rotates in the same direction, but damped. If the user has achieved the intended rotation or is stuck at the threshold, see section 3.3, and wants to move on, by pushing the bumper button of the same Razer Hydra controller, the current rotation is applied to the camera. Then the ray is shown again and the user can go on with other interactions.

5. DISCUSSION AND EVALUATION

Our approach was successful in some respects, in others we failed to find a satisfying solution within our project's time frame. The feeling of the user's movement is still not fluent and suffers from occasional small jumps in position. We suspect this could be the result of a slightly unstable marker tracking or maybe a symptom of performance issues regarding Blender. The perpetual processing of all peripheral devices in Blender could cause too high load, resulting in a non fluent rendering and overall processing, which is in turn crucial for a convincing presentation.

The Oculus Rift has turned out to be a good choice for our purpose, because besides some problems regarding the correct distortion of the images via a shader in combination with Blender, it reliably provided head turning information and a good way to display the scene to the user.

With the current functionality regarding the Microsoft Kinect and the Razer Hydra, we have elaborated two basic examples of user interaction with the environment, whereas the Razer Hydra provides a direct and according to our experience intuitive way to manipulate the scene. In contrast, the Kinect offers right now an observable video stream only.

The rather time consuming try to integrate MORSE, a simulation framework for robots based on the Blender Game Engine, was fruitless in the end and demonstrated one of our biggest challenges: Making oneself acquainted with third party code. While there exist several examples for MORSE source code, various features we needed were not covered by this examples.

Hence, it was left to us to contact the MORSE developer team and ask for help. Even though our requests were always answered extensively, it was an overall long-winded procedure, which finally was the reason to drop MORSE integration.

6. FUTURE WORK

To enable the full potential of our project in its current state, the implementation of the camera movement within the virtual environment has to be further improved. Therefor, it has to be analyzed, what exactly causes the lag during movement.

Furthermore, the camera currently mounted on the Oculus Rift features a depth sensor, which could be used to display the user's arms and hands in the simulation. There are, however, no appropriate drivers for linux existent yet. The interaction possibilities introduced by the Razer Hydra are not exhausted by far, for example a grasping mechanism could be implemented by using the position of the controllers and one of their buttons.

The visualization of the Kinect images should be sped up and increased in resolution, we consider a shader approach as promising. Multiple Kinect cameras in combination with a high resolution visualization could be used for example to observe a 3D scanned object in real time.

Adding MORSE integration to the system would dramatically extend the use for the system, so this should be targeted. The usage of the robots simulation would be much more efficient if combined with our approach, as you could visualize and explore the robots' data in a more intuitive way than on a pc monitor.

Right now our system is not very user friendly, because the different software modules have to be started separately and deploying them on a new pc is not that easy because of several software-requirements. This should be worked on in the future. The installation and usage of the system should be as easy as possible.

7. CONCLUSION

We created a virtual space by using the Oculus Rift as the display device, but unlike other applications, which only make use of the orientation of the Oculus Rift, we placed a camera on the Oculus to track markers and thus also detect the position of the user.

As an interaction device, we used the Razer Hydra to get the position and orientation of the hands of the user and thereby to manipulate the virtual space. We also make use of the Microsoft Kinect to show real world content.

We connected everything with RSB and a Cython wrapper to send the data to the Blender Game Engine, which generates the virtual view. All in all we can say, we achieved some positive, but also negative results.

The integration of the Oculus Rift, the marker tracking system, the Razer Hydra and the Microsoft Kinect into the Blender Game Engine worked very fine. But the not fluently tracking of the markers and some performance problems were big issues, which decreased the immersive feeling of our project.

Fixing these problems would though create a good system, which can be expanded to a useful tool in different areas, e.g. in facility management or for architects and furniture stores to help the customers to imagine the results.

8. ACKNOWLEDGEMENT

We would like to thank Thies Pfeiffer for providing us with the necessary software to track markers and to calibrate the camera. He also was of great help, whenever we ran into seroius problems.

9. REFERENCES

[1] Rodney A. Brooks, editor. *The Intelligent Room Project*. MIT Artificial Intelligence Lab, 1997.

[2] Andrew Graham Daniel Brooker, Toby Collett, editor. *Improving Augmented Reality Visualisation for Mobile Robot Development*. School of Engineering, The University of Auckland, Auckland, New Zealand, 2009.

[3] Burkhard C. Wuensche Ian Yen-Hung Chen, Bruce MacDonald, editor. *Mixed Reality Simulation for Mobile Robots*. Dept. of Computer Science University of Auckland New Zealand, 2009.

[4] Jonathan Foote Sagar Gattepally Don Kimber Bee Liew Eleanor Rieffel Jun Shingu Jim Vaughan Maribeth Back, Anthony Dunnigan. The virtual chocolate factory; building a real world mixed - reality system for industrial collaboration and control. FX Palo Alto Laboratory.

[5] Lubosz Sarnecki. Oculus rift support in blender game engine. http://lubosz.wordpress.com/2013/06/26/oculus-rift-support-in-blender-game-engine/, 2014. Accessed: 2014.06.30.

[6] Shean White. Interaction with the environment: Sensor data visualization in outdoor augmented reality. Department of Computer Science, Columbia University, Department of Botany, Smithsonian Institution, 2009.

[7] Z. Zhang. ”a flexible new technique for camera calibration”. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2000.

INTELLIGENT SYSTEMS PROJECT: CHEERING YOU UP IN EMPATHIC ROOMS

Christian Limberg, Benjamin Wolff

Faculty of Technology, Bielefeld University
Bielefeld, Germany

Supervisors: Felix Hülsmann, Hana Boukricha

ABSTRACT

In this work we will propose an empathic room: An adaptive room, that changes its ambience, depending on the user's mood. We present a system that automatically detects faces in image from multiple web cams and performs real-time face and emotion-detection, with respect to seven dimensions (happy, sad, surprise, disgust, fear, anger and neutral). The system is connected with an adaptive room. We introduce a concept of cheering-up the user. Depending on the emotion, the adaptive room actuates different components e.g. (sound and lights), with the aim of counteracting negative mood states.

1. INTRODUCTION

The human face is a rich and powerful source of information. In face-to-face communication it reflects our feelings and emotions, thus it is often called the window to the soul. Therefore, the use of facial expressions in Human Computer Interaction gains more and more interest.

- How can an intelligent system recognize and use these information?
- How can an intelligent system adapt to the user's emotion?
- Which features can be extracted for processing?

The key questions in our work are: How can an adaptive room respond to the user's emotion? How can an adaptive room keep the user happy, by supporting positive emotions? How can an adaptive room cheer up a user with negative mood?

In Winter-Term 2013/2014 we focused on how supervised learning techniques can be applied to detect emotions in video-streams and developed a software prototype. In Summer-Term 2014 we focused on developing "Cheering-Up"-concepts, actuating components in the intelligent room, improving classification results and enhanced software functionality.

Most related work focuses on emotion classification and detection. Bartlett et al. [1] used different machine learning methods for a fully automatic detection of facial expressions. This work is based on the recognition of facial actions (FACS). They achieved best results with gabor filters as feature extraction and classifiaction with Support Vector Machines (SVMs). Zhou et al. [2] propose a framework, named "AmE", for building emotion-aware applications, with the aim to increase the user experience (UX). Nevertheless the paper only adresses technical challenges and not in which way a system can change its ambience depending on the user's mood.

Whitehill et al. [3] go a step further: They use automatic facial expression for intelligent tutoring systems. In this context, facial expression can give an automated feedbeck, e.g. the student's motvation or the estimated level of difficulty perceived by the student.

It is worth noting, that we have not found any system that implements an empathic room, reacting on the user's mood.

2. CHEERING-UP CONCEPTS

Whatever we do, the environment around us has a great influence on our mood. Feng Shui masters knew this already for almost thousands of years ago. Nowadays small and powerful systems allow us to adapt the environment seamlessly and in real-time.

In the following we propose a concept, that counteracts negative mood states and supports positive mood. In short: A concept of how "Cheering You Up In Empathic Rooms" can be achieved.

Ekman et al. [4] propose six discrete basic emotions: anger, disgust, fear, happiness, sadness and surprise. We use these basic emotions as a starting point. It is important to distinguish between emotions which relate to negative and positive mood. Emotions reflecting negative mood are: anger, disgust, fear, sadness. The only emotion reflecting positive mood is happy. Surprise is an emotion, of very short duration and not necessarily related with positive or negative mood. Thus it will not be used in our concept. We decided to use a general mapping from detected emotion to room adaptation (see figure 1): The camera image is preprocessed and the user's emotion will be detected. Next, a mapping from detected emotion to desired effect on the user is done. For angry, disgust and afraid we aim to calm the observed user. For sad emotion our goal is to cheer-up. If the user shows happy emotions, we aim to support the current state. To keep the system unobtrusive, for neutral emotion no actions will be performed. Next, we have to specify, how the desired adaptions can be achieved. We decided to use findings based on color and music psychology.

2.1. Audio Stimuli

In our system, we use a set of audio stimuli, introduced and evaluated by Eerola et al. [5]. The stimuli set consists out of 110 film music excerpts, with a length of 15 seconds. In a listening experiment one half were evaluated with respect to the perceived emotion (anger, fear, sadness, happiness and tenderness), and the other half to the extremes of three bipolar dimensions (valence, energy arousal and tension arousal). Each label set contains examples with moderate and high intensity. According to our concept we chose the following stimuli:

- calm: tender moderate/high, valence positive high
- cheer-up: happy high, energy positive high
- support: happy moderate, energy positive moderate

2.2. Visual Stimuli

The selection of visual stimuli is based on Naz et al. [6]. Naz et al. evaluated these stimuli in terms of their emotional responses. These stimuli were referenced to the Munsell Color System and can be reproduced. According to the results blue (10B) evoked a feel of calm to the subjects. Yellow (7.5Y) and green (2.5G) were highly related with an positive emotional association. Blue-green (5BG) evoked a moderate positive feel. figure 2 illustrates how these colors are mapped in our system. It shows the mapping from emotion to concrete selection of colors (coded in Munsell Color System).

3. CUEP SYSTEM

3.1. Overview

The basic system is illustrated in figure 3. It starts with a potential user, who is located in an adaptive room and observed by one or more cameras. For reasons of flexibility our system relies on IP-cameras. That allows an easy adding of cameras. Our application receives the camera signal frame by frame. Next, our application performs face detection, face tracking and facial feature detection. The facial feature points are coordinates of 66 points (displaying the position of eyes, eye contours, eyebrows, lip contours, tip of the nose, ...). This is the input for our real-time emotion detection. Basically we use a one-vs-all two-class SVM with probability estimates for every basic emotion (anger, disgust, fear, happiness, sadness and surprise). For room adaptation we use a moving average, with freely selectable frame-width and threshold. Depending on the detected emotion, the room control actuates music or audio stimuli.

3.2. Hardware

The CUEP-System runs on a laptop (INTEL i7 q720, 4GB RAM) with no special needs. It communicates with IP-cameras, which you can add or remove in the graphical user interface. We use IP-cameras with 720p signal format. A higher resolution is possible but requires more cpu-usage for the CUEP-System to process.

For sound output we use speakers located in different places in the adaptive room. The light output is provided by four RGB-spotlights (Stairville LED PAR 64), controlled with a peperoni rodin 1 “USB to DMX” converter [7].

3.3. Emotion Recognition

3.3.1. Software

In our work, we used Luxand FaceSDK [8], an application programming interface, which provides methods for face detection and face tracking. Furthermore FaceSDK easily allows to extract facial features. The facial features consists of 66 2D-Points, placed on defined locations in the face. We used this method for feature extraction.

For supervised learning we decided to use support vector machines with radial basis function kernel. We used libsvm [9] for training and online-classification.

In our previous work (Winter-Term 2013/2014) we decided to use an AU-based emotion detection. The concept of action units (AUs) is based on the FACS manual [10]. Facial muscle contraction produces facial movements. Ekman defined action units. An action unit represents a facial movement, produced by contraction and/or relaxation of one or more facial muscles.

For this purpose we used a linear one-vs-all two-class SVM for each AU detection. For our System, we used 13 Action Units (AU1, AU2, AU4, AU5, AU6, AU7, AU9, AU10 ,AU12, AU15, AU20, AU24, AU25). Although we achieved acceptable performance with the AU based approach, it had a number of disadvantages:

- There are hardly any databases with action unit labelled images
- In the database, we used for training (CK+ [11]), there were not enough positive samples for some AUs

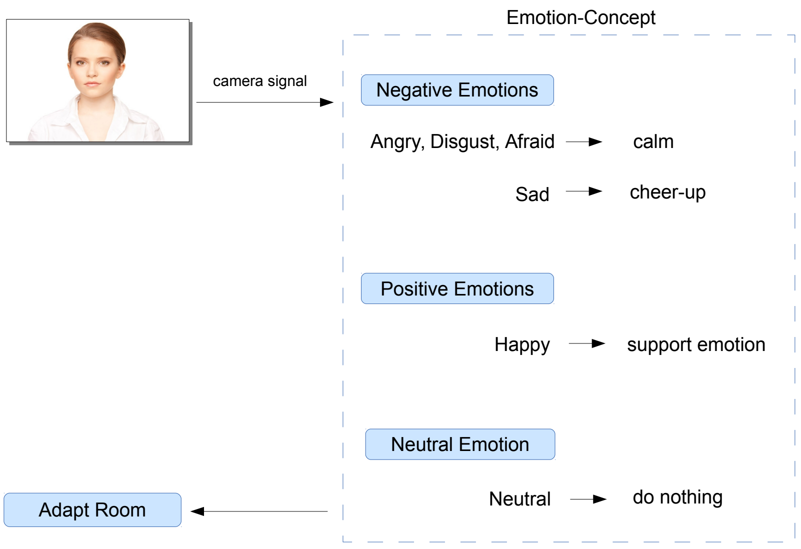


Figure 1: Emotion-mapping

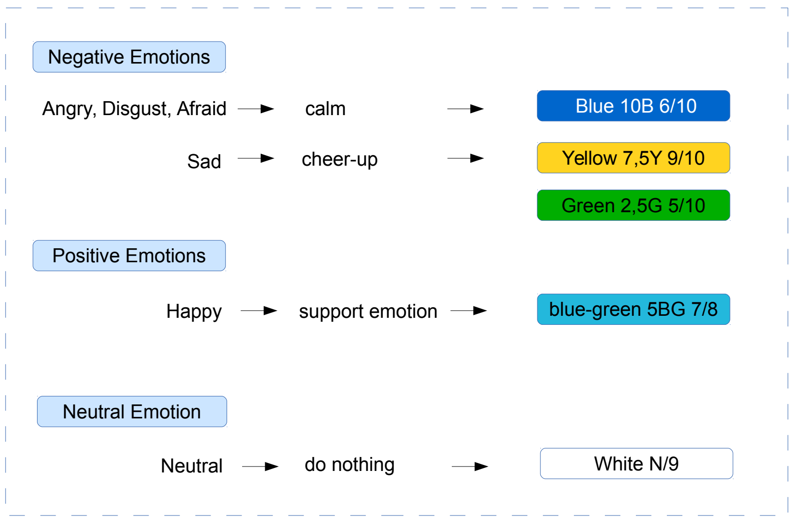


Figure 2: Light mapping

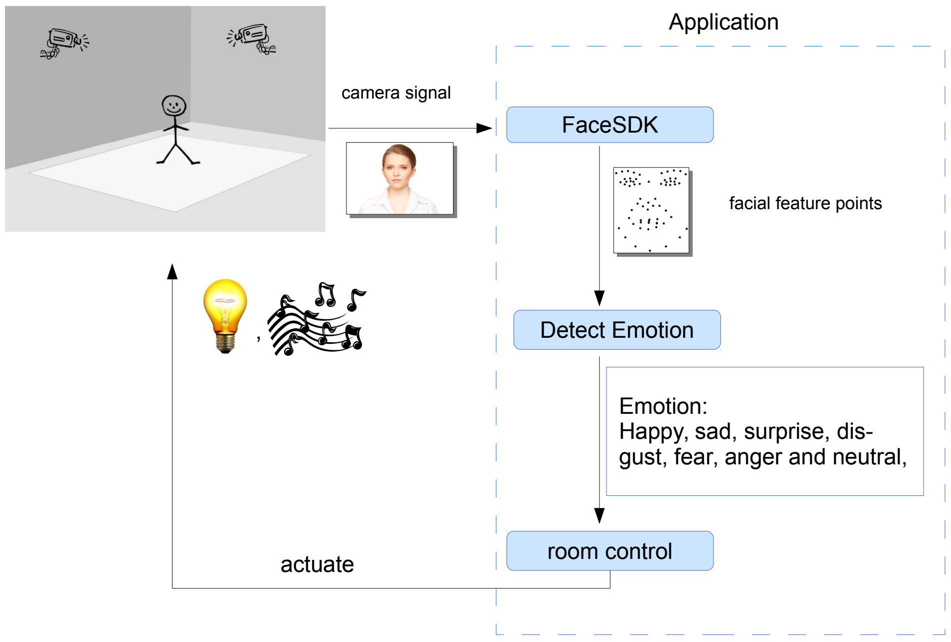


Figure 3: Illustration of the system setup

- Mapping from AU to emotion delivers a binary classification. It lacks probability estimates, because there is no weighting, how much an AU contributes to an emotion.

To overcome these issues we decided to switch to direct emotion classification.

The training procedure ist illustrated in figure 4.

3.3.2. *Dataset*

For training, we use emotion labelled images from two databases:

- The extended Cohn-Kanade-Database (CK+) [11]
- Karolinska Directed Emotion Faces (KDEF) [12]

We use both to train our SVMs for emotion detection. The CK+-database contains 593 sequences from 123 subjects with a resolution of 640x490 pixels. We used a one-vs-all two-class SVM for each emotion. All sequences are from neutral face to peak expression. 327 of the sequences were emotion-labelled (basic emotions and neutral). Only 327 of 593 sequences are emotion-labelled. The peak-frame from sequences containing the emotion were used as positive samples. All neutral frames and peak-frames not containing the emotion were used as negative samples.

The KDEF-database contains images from 70 amateur actors (35 female, 35 male) posing 7 different emotions (basic emotions and neutral) from 5 different angles in 2 series. For our training set we only used pictures with straight angle. This results in 980 samples.

We combined these datasets to one training dataset.

3.3.3. *Preprocessing*

Before initiating SVM training, we preprocessed the training dataset: FaceSDK was used to extract facial features. We decided to resize all facial features to an inter-ocular distance of 77px. All resulting features were scaled in the range [-1,1].

3.3.4. *Training Procedure*

We decided to use one-vs-all two-class SVMs with probability estimates, one for each emotion. For probability estimates, we use a confidence value, which represents distance to the seperating hyperplanes We used radial-basis-function (RBF) Kernel for our SVMs. Since our training set is imbalanced, we adusted the class weights. In order to find the best parameters (C, γ), we used an exhaustive grid search with 5-fold cross validation.

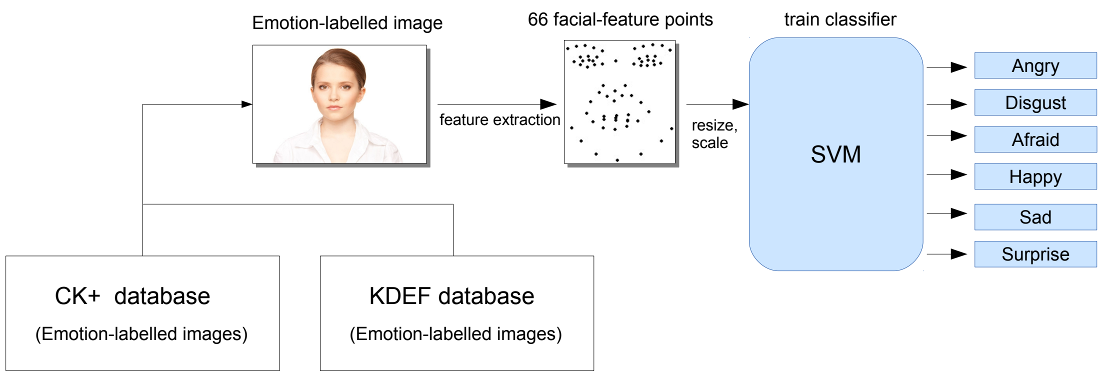


Figure 4: Training procedure

3.3.5. Evaluation

Generalization performance of emotion detection was tested using 5-fold-cross-validation on the training dataset. The results are shown in Table 2. From these results we expect very good recognition rates for the emotions happy, surprise and disgust and good recognition rates for angry, afraid and disgust. In order to assure reliable results, a further evaluation with a database containing emotion labelled face images, should be performed. Table 2, shows the results from our AU-based classification. We used the CK+-Database as training set and the KDEF-Database as test-set. The results are shown in Table 1 and compared to an average human observer. The performance of an average human observer is based on a validation study from Goeleven et al. [13].

Emotion	Performance
Angry	90.94%
Disgust	94.06%
Afraid	90.39%
Happy	98.41%
Sad	90.64%
Surprise	96.14%

Table 2: Performance (5-fold cross validation) of direct emotion detection

4. DISCUSSION

The robustness/effectiveness of our system strongly depends on adequate lighting conditions, the speed of head movements, camera quality, and the camera-to-face distance. To increase the systems reliability, high resolution cameras are recommended. Nevertheless our system achieves a reasonable detection rate, that partly even outperforms an average human obeserver.

In the course of the project, we experienced that our method for feature extraction (FaceSDK) has several disadvantages. Though it is easy to use, the localization of facial features is too imprecise for a crucial task like emotion detection. Furthermore FaceSDK will only work properly, if the subject don't wears a beard or eyeglasses. We also experienced problems in facial feature detection, if the participant's mouth is wide open. For our system, we combined two datasets: KDEF and CK+.

Both provide images from subjects without eyeglasses and beard. Hence, subjects with beard and/or eyeglasses will experience a performance loss in emotion classification. Our system would benefit from more training data. However, suitable databases with emotion labelled data are not easy to find. It is to say, that our system can only detect frontal faces. This can be compensated by using multiple cameras.

But nevertheless: if the facial feature detection problems could be solved, e.g. with another detection-library or improvements in Face-SDK, the CUEP-System is a really reliable and innovative room component, that could be established in modern households, to assist the feelgood factor of residents.

Emotion	Human Observer	CUEP-System (AU-based)	CUEP-Performance
Neutral	62.64%	90.00%	+27.36%
Happy	92.65%	92.90%	+0.25%
Sad	76,70%	72.70%	-4.0%
Surprised	96.00%	95.70%	-0.3%
Disgust	72.17%	53.60%	-18.57%

Table 1: Performance of AU-based emotion detection on KDEF-Database compared to an average human observer (based on a validation study from Goeleven et al. [13])

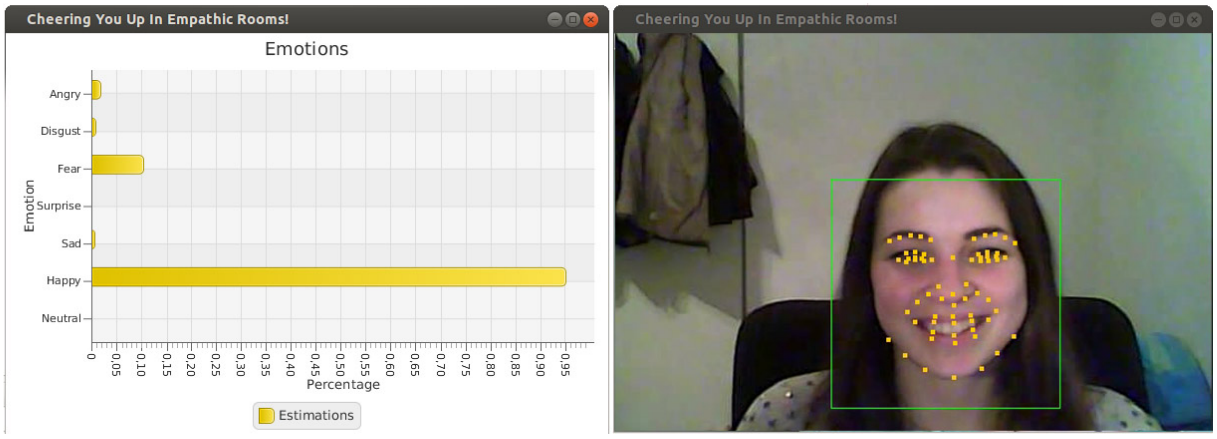


Figure 5: Graphical User Interface of CUEP-System. The emotion barchart on the left is updated in real time and displays the estimated emotion probability.

5. CONCLUSION AND FUTURE WORK

In this paper, we presented an empathic room: An adaptive room, that changes its ambience, depending on the user's mood.

This goal was achieved by a system that:

- performs real-time face detection with IP-cameras
- performs emotion-detection based on facial features (AU-based or directly) and SVM
- implements cheering-up mechanisms
- actuates music and lights

A screen shot of the graphical user interface is shown in figure 5. We showed that emotion detection is possible with two different approaches: AU-based(two-step) and directly (one-step). A performance test for the second approach is still pending. Our actual results suggest, that it delivers higher classification performance. One reason for this might be, that we used more training samples. It is unclear, if AU-based classification would achieve same results, with more training samples. Though AU-coded images are hard to find. However AU-based classification has one advantage. Since it only represents muscle activations, it is more flexible and can be interpreted in many ways.

As future work, we plan to determine performance of direct emotion detection. We plan to use the Radboud Face Database [14] as test-set. We requested access for Radboud Face Database, that was not granted yet.

Furthermore it would be nice to include more output classes to the system. In our system we decided for playing music and changing light, but there are many different ways to affect emotion in an empathic room. Our system could control a micro controller to handle other devices. Further efforts should be made, to determine how the ambience affects the person's emotion.

It could also be interesting to implement a tracking feature in the CUEP-system. This enables to create a user database with preferences in music or other settings, to even better fit in and handle the needs of the users.

6. REFERENCES

[1] Marian Stewart Bartlett, Gwen Littlewort, Mark Frank, Claudia Lainscsek, Ian Fasel, and Javier Movellan. Recognizing facial expression: machine learning and application to spontaneous behavior. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 568–573. IEEE, 2005.

[2] Jiehan Zhou, Changrong Yu, Jukka Riekki, and Elise Kärkkäinen. Ame framework: a model for emotion-aware ambient intelligence. In *Proceedings of the second International Conference on Affective Computing and Intelligent Interaction (ACII2007): Doctoral Consortium*, 2007.

[3] Jacob Whitehill, Marian Bartlett, and Javier Movellan. Automatic facial expression recognition for intelligent tutoring systems. In *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW'08. IEEE Computer Society Conference on*, pages 1–6. IEEE, 2008.

[4] P Ekman and WV Friesen. Pictures of facial affect, 1976. *Consulting Psychologists, Palo Alto, CA*.

[5] Tuomas Eerola and Jonna K Vuoskoski. A comparison of the discrete and dimensional models of emotion in music. *Psychology of Music*, 2010.

[6] KAYA NAz and Helena Epps. Relationship between color and emotion: A study of college students. *College Student J*, 38(3):396, 2004.

[7] Peperoni Rodin 1 "usb to dmx"-adapter. http://www.luxand.com/facesdk/.

[8] Luxand FaceSDK. http://www.luxand.com/facesdk/.

[9] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

[10] P Ekman, WV Friesen, and JC Hager. Facs manual. *A Human Face*, 2002.

[11] Patrick Lucey, Jeffrey F Cohn, Takeo Kanade, Jason Saragih, Zara Ambadar, and Iain Matthews. The extended cohn-kanade dataset (CK+): A complete dataset for action unit and emotion-specified expression. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*, pages 94–101. IEEE, 2010.

[12] Daniel Lundqvist, A Flykt, and A Öhman. The karolinska directed emotional faces. *Stockholm, Sweden: Karolinska Institute*, 1998.

[13] Ellen Goeleven, Rudi De Raedt, Lemke Leyman, and Bruno Verschuere. The karolinska directed emotional faces: a validation study. *Cognition and Emotion*, 22(6):1094–1118, 2008.

[14] Oliver Langner, Ron Dotsch, Gijsbert Bijlstra, Daniel HJ Wigboldus, Skyler T Hawk, and Ad van Knippenberg. Presentation and validation of the radboud faces database. *Cognition and Emotion*, 24(8):1377–1388, 2010.

[15] Ellen Goeleven, Rudi De Raedt, Lemke Leyman, and Bruno Verschuere. The karolinska directed emotional faces: a validation study. *Cognition and Emotion*, 22(6):1094–1118, 2008.

INTELLIGENT SYSTEMS PROJECT: COOKING WITH A ROBOT 1 – THE COGNITIVE KITCHEN-NET

A. Dreyer, T. Schürmann, H. Kaufhold

Faculty of Technology, Bielefeld University
Bielefeld, Germany

Supervisors: C. Dreyer, M. Pohling, S. Rüther, S. Wrede

ABSTRACT

This paper presents the back end of the Cooking with a Robot (CWAR) project, an interactive cooking guide. Consisting of a workflow representation of recipes in Business Process Model Notation and a connection to the kitchen devices via a hardware gateway from Miele, CWAR leads the user step by step through a recipe. The user is supported by automatic error detection (e.g. check temperature of the oven) to prevent failures. The application provides an interface for the use of different user interfaces. A user study was performed to evaluate the system. The results show that the basic concept works and a recipe can be presented suitable in a workflow representation.

1. INTRODUCTION

Cooking ones own food can be a fun and relaxing activity during the usually stressed day as well as save ones money and improve ones health. However, in our modern society a lot of people do not cook for themselves anymore or have only a very small variety in the recipes they cook. We assume that a big proportion of these people do not cook for themselves because they simply might not know how to cook. Even if these people want to learn cooking, they are often deterred by the complexity of more advanced recipes and thus might not have the time or motivation to try them out in order to improve.

In order to encourage less experienced cooks to cook more often or try out new recipes, we want to provide them with an interactive kitchen, helping the user during the cooking process. The multimodal and context-aware assistive system uses knowledge about known recipes and information about the status of the kitchen hardware, provided by modern kitchen devices, to guide the user, step by step, through a recipe and informs the user about potential errors, e.g. not preheating the oven. Information about the different steps in a recipe are provided visually in form of a virtual cookbook, which can be projected onto the wall behind the kitchen counter, as well auditory by reading the steps to the user as well. In order to limit hesitation in using the system, the user interacts with the system using voice and gesture tracking. Unlike with physical input devices, the user does not need to fear soiling the system when using it after dirtying his or her hands. The voice control also allows the user to manipulate the system while moving around the kitchen freely. We also employ a virtual animated representation of the robot Flobi, providing a concrete entity the user can interact with.

As part of the seminar “Intelligent Room” at Bielefeld University, three groups have worked at this project. For this the project has been split into three main parts as can be seen in section 3. Our group worked on the workflow model and the hardware interface, that are described in section 5. The workflow model is used for modelling the recipe in a way a machine understands. The hardware interface is used for communication with the Miele@Home system, which is described in section 4, and also implements a simulation for use without the actual devices.

2. RELATED WORK

As an everyday example the kitchen scenario is an often chosen topic in research. PersonalChef [1] and CounterActive [2] are projects that want to help the user within the cooking process with background information and tutorial videos. The setups of these projects provide displays and projections on which the user can touch to interact with the application. The single steps of the recipes are shown step by step and further information, for example on ingredients, can be invoked. We want to help the user cooking as well but next to the single steps we do not have further information like videos for the user. In contrast to the presented projects we use in cooperation with the other CWAR groups a leap motion and voice control. As PersonalChef and CounterActive our system can guide through only one recipe at once. A similar system called Cooking Navi [3] allows the user to finish two dishes in parallel.

Projects that uses voice control like us are for example eyeCOOK [4], which additionally to speech commands uses eye-gaze and provides timer functions, and Kochbot [5], that deals with german commands and recipes and is a portable application for smartphones and tablet computers. The complete CWAR system is not able to run on a smartphone because of the simulated robot in the cookbook that needs a high amount of computing.

Although there are several cooking assistant technologies as far as we know there is no project yet that includes status information from kitchen devices the way we do. An application called SmartKitchen [6] uses the serv@Home appliance from Siemens as well as cameras and other devices to record recipes, annotate it and post it in the internet but no checks of user activity are done. At CeBIT the team of the Kochbot presented an integration of kitchen devices, similar to our approach in checking the users actions for errors, with the Universal Remote Console technology (OpenURC[1]) but at this time there does not seam to be a publication.

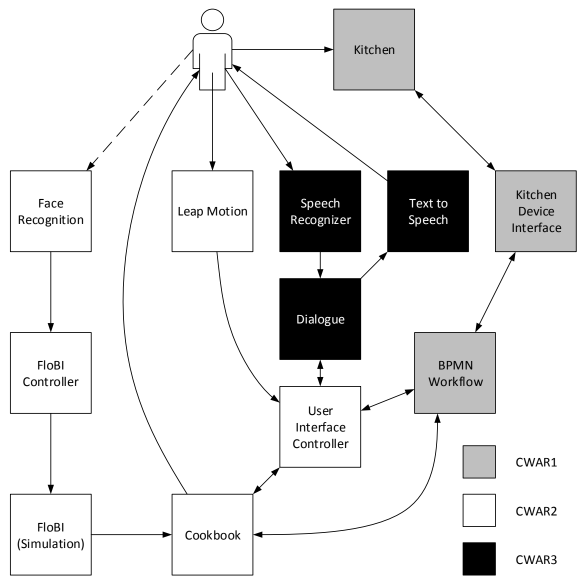


Figure 1: Components of the overall system: The back end is coloured grey and consists of the components described in this paper. The white coloured boxes represent the virtual cookbook and the leap motion control. The black coloured part is the speech recognition system.

[1]see `www.openurc.org`

3. SYSTEM DESIGN

The interactive kitchen consists out of three main parts as shown in figure 1. One part is the cookbook, the incorporation of the FloBi-robot and the gesture control and synchronisation between gesture and voice control for the back end. For further information about these parts we refer to the paper of group CWAR-2 [7].

Another part is the voice control, recognizing users voice commands and providing auditory information to the user. For more information about the dialogue components we refer to the report paper of group CWAR-3 [8].

The remaining parts are the workflow model and the hardware interface, which will be described further in this paper. These parts represent the back end and therefore the complete control flow.

4. HARDWARE COMPONENTS

The kitchen devices are connected with the Miele@Home system of the Miele company. This system allows to connect different devices via power line communication. With the Miele@Home gateway states of the devices can be read and selected actions (e.g. turning on the light of the hood) can be performed. Further more one can listen to update information about state changes of the devices. At the moment the supported devices are a hob, an oven as well as a hood. Other devices can be integrated easily.

5. SOFTWARE COMPONENTS

The software consists of three parts: a workflow model that holds the recipe, a hardware interface for communication with the kitchen and a simulation to work with if the real kitchen devices are not usable. In the sections 5.2 to 5.4 every component is described in detail. For the communication between these three parts and with the software of the other groups we used the RSB software that is described in section 5.1.

This project needs the following requirements to be met:

- Java 7.0 or higher
- RSB 0.9.4[2]
- spread 0.9, see RSB installation instructions[2]

5.1. RSB/RST

The Robotics Service Bus (RSB) is used to transport the Robotics Systems Types (RST) from one component to another. RSB is a “message-oriented, event-driven middle-

[2]`http://docs.cor-lab.de//rsb-manual/0.9/html/index.html`

ware aiming at scalable integration of robotics systems in diverse environments”[2]. Via RSB simple data types like strings as well as RST can be send. The Robotics Systems Types Repository “contains type specifications for robotics and cognitive systems and associated conversion code for different data types and programming languages.”[3] The within this project defined RST are shown in figure 2. For further information about RSB see [9].

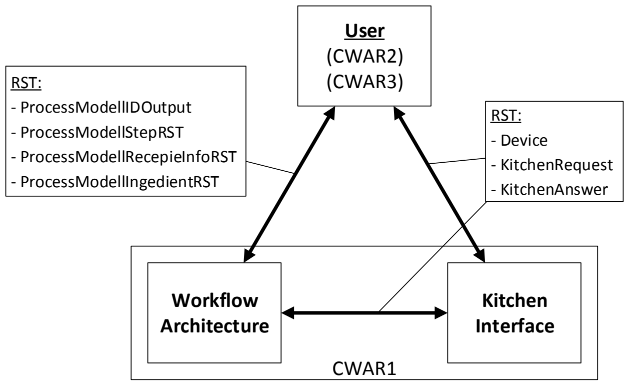


Figure 2: Communication between different project parts with RSB

5.2. Workflow model

The hypothesis of this project is that coordination of tasks can be achieved by utilizing workflow models in a flexible cognitive software architecture. Business Process Model and Notation (BPMN) 2.0 is used for this computer programme. BPMN is a standard for business process modelling providing a graphical notation for specifying business processes in a business process diagram. For the process used here see figure 3. An existing architecture for coordination of human-machine interaction inspired to utilise Activiti as process engine [10].

The basic idea was to utilize the symbols of BPMN to write a workflow representation of a recipe. A symbolic representation is processed by an workflow engine, which has some common features to simulations of state machines. The workflow engine verifies the following conditions in every step of the process diagram:

- Is the current state valid?
- Is the user permitted to execute the task?
- Is every execution condition met?

[3]http://docs.cor-lab.de//rst-manual/0.9/html/index.html

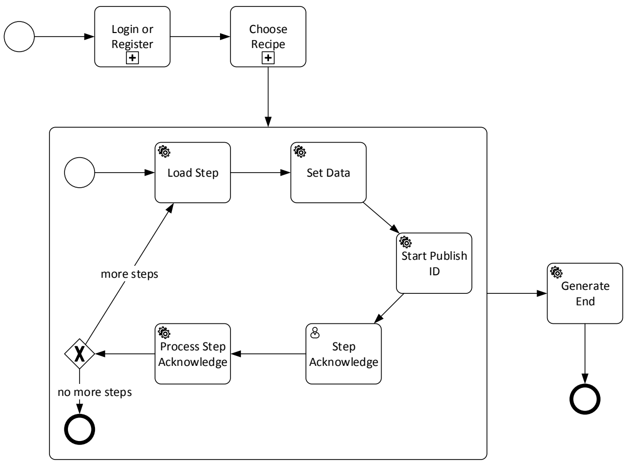


Figure 3: BPMN representation of generic recipe workflow: This is a process executed for each recipe. Starting with an optional “Login or Register”the cooking process starts with the choice of the recipe. The sub-process following contains a generic workflow mapping every selectable recipe to a set of steps the process engine has to accomplish. Starting with load and store of step relevant data into the model, the respective StepID will be published later. After publishing the ID, model checks for certain step dependent conditions (user and/or hardware). These conditions are then evaluated and if there are more steps available the cycle restarts.

We have implemented a generic recipe workflow in BPMN, that can be found in figure 3. This workflow is a simplified version of any selectable recipe and contains a set of steps the engine has to accomplish in every recipe step.

The generic workflow is decoupled from the level of detail provided in the recipe steps, to allow the use of different complex recipe steps on basis of the same workflow structure. At the same time the process engine has to use a fast parallel monitoring of device status to track the cooks progress in the recipe steps. Monitoring and user control can be supervised by any state machine simulation on hand, but the Activiti library provides additional expandability, flexibility and the possible utilising of a user management which may be difficult to implement in a state machine.

For the evaluation, test data was added to the process model by utilising an hand written recipe editing software.

We chose the hamburger recipe for the study, because it utilises every device provided by Miele@Home.

The interprocess communication, as could be seen in figure 2, is controlled by a detached cyclic process using asynchronous RSB message interface. In every single cycle a step of a recipe is illustrated, necessary data is generated, processed and verified with the kitchen hardware. If a problem occurs during the verification process, the model freezes and waits for an appropriate cook action. Examples may be a low or a too high temperature reading on a hob or a serious incident like a switched-off or even broken oven. This information is provided via the kitchen interface.

Based on the study recipe the sequence is now described with few more details: For the test, it is assumed the user has already chosen the hamburger recipe and therefore has completed the part of “Choose Recipe”in figure 3. The first step of the burger recipe is loaded. In this case the instruction is: “Please preheat the oven up to 175 degrees Celsius.”

Important information such as the temperature to preheat the oven is saved. After the sufficiency of the data is verified, the current ID of the step is broadcast via the RSB interface mentioned above. The process model now waits until certain physical and task specific conditions described in the step are met. These conditions could be reaching a certain device temperature level, that is provided by Miele@Home gateway via the hardware interface, or a required user input. In the instruction mentioned in the paragraph above the engine is waiting for RSB messages containing information whether the oven has reached 175 degrees Celsius or not. After the cook acknowledged the current step a search for more steps is performed. If there are no further steps the recipe is completed. If there are more steps, the software repeats the cycle.

5.3. Hardware interface

This interface reads the information given by the Miele@Home gateway. For this task it parses the XML file from the gateway and builds an device object out of the given information. These device types can be send to other components using RSB (see section 5.1). It also deploys an answer if any requests were given, for example turning the light on or switching the devices off.

Another aspect of this interface is to verify that the devices work properly, or cannot be used. For this task the interface uses a test system similar to “build in test equipment”(BITE). Since it is not possible to add any more hardware devices the interface examines the devices state and functionality at software level on every start-up. So it checks for adequate states of the devices, and if a device is not usable then it prints an alert message. This also applies for abnormal data. Furthermore it stops the device from being addressed through the interface as long as it is not available.

5.4. Simulation

The simulator is an independent part of the hardware interface. It is used for testing if the real kitchen devices are not usable. The simulator provides a hob, an oven and the light of the cooking hood. It uses the same communication interface as the hardware interface with the following differences to the real devices:

- The oven heats itself three times faster than the real one.
- Only one single cooking plate is provided at the current status.
- It provides an instant heat function for the hob and the oven.
- If a temperature changes (heating or cooling) it notifies the process model every three seconds of the actual state (the Miele@Home system informs you at once). This is used to keep the network traffic and the logs at a minimum.
- The devices can also be switched to on, using the provided user interface.

6. INTERACTION EXAMPLE

Interaction with our components can be seen in our earlier interaction video[4]. This animated video gives an idea how the system works.

A video showing the entire system at work will soon be available on the course website[5].

7. EVALUATION

For the evaluation eleven participants, three women and eight men in the age of 23 to 34, were asked to cook a hamburger following the instructions of the CWAR system. We chose the hamburger recipe because it do not take too much time and every device in the kitchen is used. The participants had different experience in cooking processes as visualised in table 1, and were all familiar with the use of computers. Most of the participants work in the Citec building and are colleagues. Some others are students. With our task non of them was familiar. They were invited to cook hamburgers in a real kitchen with the Miele@Home devices. They started the cooking process with a voice command and were guided step by step by the system. A step was finished with a voice or gesture command. After finishing the cooking process the participants filled out a survey to outline their experience while cooking with CWAR.

[4]http://www.techfak.uni-bielefeld.de/isy-praktikum/WS13SS14/CWAR-1/media/video.mp4

[5]http://www.techfak.uni-bielefeld.de/isy-praktikum/WS13SS14/

Table 1: Cooking experience of the eleven probands

	no experience						much experience
cooking	0	2	0	2	5	1	1
difficult recipes	3	4	2	0	2	0	0

The different CWAR groups were interested in different aspects. Our group focused on the usability of the workflow model and the performance of the hardware interface.

7.1. Number and apportionment of steps

We wanted to find out if the partitioning was adequate or whether the cooking steps were too many and therefore the tasks were too small or whether the tasks were to big.

The answers of the participants are shown in figure 4. According to the survey 63,64% of the participants found the number of steps appropriate, 18,18% would like to have more steps and 18,18% would like to have even less steps.

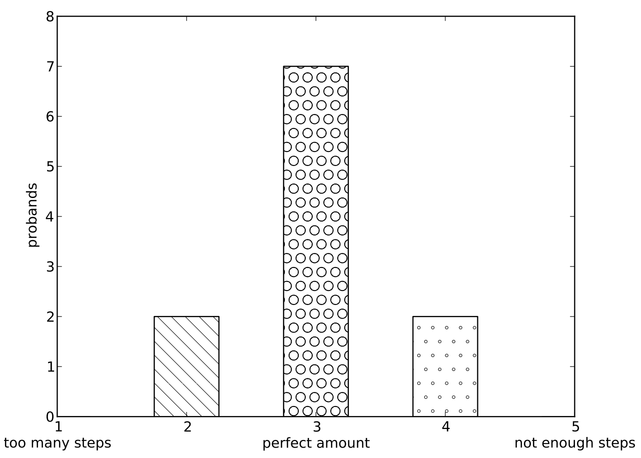


Figure 4: Answers of the probands to the question “Was the number of steps appropriate?”

Then we asked the probands what they think about the apportionment of steps. 45,45% of them said that the apportionment is very expedient as one can see in figure 5. Three participants would like to have a bit more information in each step while one indicated that he missed much information in the single steps.

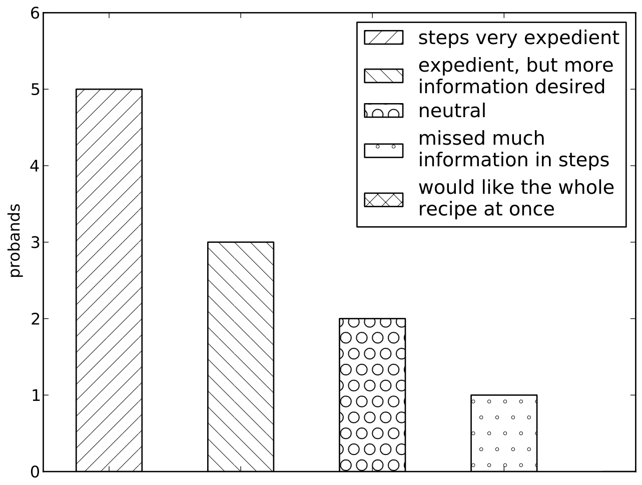


Figure 5: Answers of the probands to the question “Was the apportionment of steps expedient?”

7.2. System answer speed

Our purpose was to identify the users feeling while using the system according to the reaction speed of the system. We wanted to know whether the system answer is fast enough or the user have the expression that the system does not work well. 36,36% of participants were content with the speed of our system as shown in figure 6. 27,27% felt that it was slightly too slow while three probands said the system was too slow. One person decided that the system reacted too fast to his input.

8. DISCUSSION

The study shows that the basic idea of modelling the recipe in these particular steps is proved successful. Most of the participants of the study liked the amount of steps and the information content. So the hypothesis that the workflow model is capable of providing an adequate model for recipes is proved. It is possible to supervising the devices too. So the goal to show this is satisfied although still some problems were encountered.

The reason why many probands experienced the system as too slow may be found in the user interface components. An evaluation of the RSB protocol shows that the transmission needs only a few milliseconds to send and receive the information. Sometimes delays were caused by the Miele@Home gateway as it is only a prototype yet.

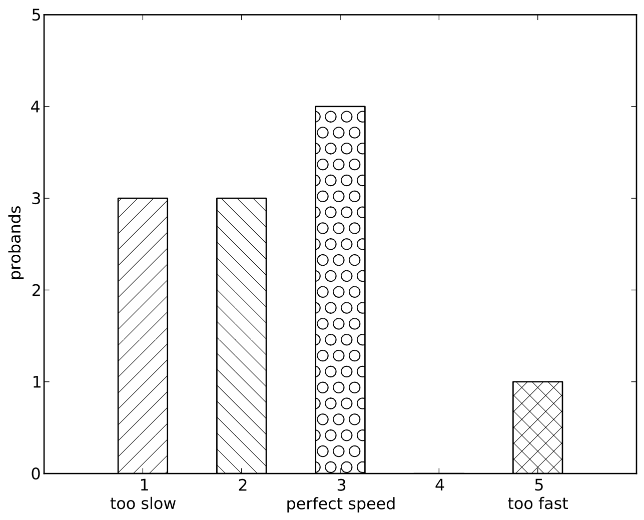


Figure 6: Answers of the probands to the question "Did the system answer your inputs fast enough?"

For future work first of all some control functions should be added. An important point would be to provide the possibility to undo a step the user acknowledged too early. Another idea is to implement parallel cooking processes, first within one recipe and later on for cooking several dishes at once. Both these ideas will need a redesign of the workflow model.

Further one could implement a recipe editor to edit and create own recipes. It is also possible to implement an user account for different users, to provide individual help. Some unexperienced user could get more information, while the more experienced user could cook more advanced recipes or cut off some of the help the system provides. It is possible to create an achievement system to track the users advancing in its cooking skills and to provide a kind of competition to stimulate the user to have a go at the next level.

Another idea for the future is to implement a rating system for the meals. This could be in form of traffic lights to help users cooking more healthy meals. This would be helpful for a diet or for other goals achieved while eating, like cooking meals which containing protein to aid in sports etc. For this case it is also possible to implement a recipe chooser which proposes recipes to the user according to his eating characteristics and goals.

9. CONCLUSION

The goal to develop a system for maintaining recipes and device information for an intelligent kitchen to aid the user were achieved.

The recipe was successfully represented in a workflow model using BPMN. With the information from the Miele@Home gateway device the cooking process can be controlled and errors, like a too low temperature of the oven, can be detected. After an integration with the user interface groups, with which out system communicates via RSB messages, a study was made in which the participants cooked a hamburger with the help of the CWAR system, that guided the user step by step through the recipe. With voice and leap motion control the user could finish a step and go on with the next one.

This study shows that the basic concept works. The participants mostly liked the partitioning of the cooking steps. The tasks in the single steps were not too big. In the future the project should be worked on. Some participants said that the system answer to the input was too slow. This is one aspect that can be improved. Further more a number of additional features can be imagined like a login function to hold preferences of different users.

10. ACKNOWLEDGEMENT

We would like to thank all the supervisors that helped us during the past semesters, especially we would want to say a special thank you to Andreas Kipp for his invaluable help with the study and Marian Pohling with his help with the Miele@Home gateway. Furthermore, we would like to thank all the participants of the study, who greatly helped us identify problems of our system.

11. REFERENCES

[1] S. Mennicken, T. Karrer, P. Russell, and J. Borchers, "First-person cooking: a dual-perspective interactive kitchen counter," *Proceedings of the 28th of the international conference extended abstracts on Human factors in computing systems - CHI EA '10*, 2010. [Online]. Available: http://dx.doi.org/10.1145/1753846.1753992

[2] W. Ju, R. Hurwitz, T. Judd, and B. Lee, "Counteractive," *CHI '01 extended abstracts on Human factors in computing systems - CHI '01*, 2001. [Online]. Available: http://dx.doi.org/10.1145/634067.634227

[3] R. Hamada, J. Okabe, I. Ide, S. Satoh, S. Sakai, and H. Tanaka, "Cooking navi," *Proceedings of the 13th annual ACM international conference on Multimedia*

- *MULTIMEDIA '05*, 2005. [Online]. Available: http://dx.doi.org/10.1145/1101149.1101228

[4] J. S. Bradbury, J. S. Shell, and C. B. Knowles, "Hands on cooking," *CHI '03 extended abstracts on Human factors in computing systems - CHI '03*, 2003. [Online]. Available: http://dx.doi.org/10.1145/765891.766113

[5] U. Schäfer, F. Arnold, S. Ostermann, and S. Reifers, "Ingredients and recipe for a robust mobile speech-enabled cooking assistant for german," in *KI 2013: Advances in Artificial Intelligence*, ser. Lecture Notes in Computer Science, I. J. Timm and M. Thimm, Eds. Springer Berlin Heidelberg, 2013, vol. 8077, pp. 212–223. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-40942-4_19

[6] M. Schneider, "The semantic cookbook: sharing cooking experiences in the smart kitchen," *3rd IET International Conference on Intelligent Environments (IE 07)*. [Online]. Available: http://dx.doi.org/10.1049/cp:20070401

[7] J. Hemminghaus, T. Schodde, and J. Pöppel, "Cooking with a robot 2 - a virtual flobi cookbook," in *ISY Proceedings Summer Term 2014*. CITEC, 2014.

[8] M. Chromik, P. Pekala, and A. Vorwerg, "Cooking with a robot: The dialogue system," in *ISY Proceedings Summer Term 2014*. CITEC, 2014.

[9] J. Wienke and S. Wrede, "A middleware for collaborative research in experimental robotics," ser. IEEE/SICE International Symposium on System Integration (SII2011). IEEE, 2011.

[10] S. Rüther, T. Hermann, M. Mracek, S. Kopp, and J. J. Steil, "An assistance system for guiding workers in central sterilization supply departments," ser. Proceedings of the 6th International Conference on Pervasive Technologies Related to Assistive Environments. ACM, 2013, pp. 31–38.

INTELLIGENT SYSTEMS PROJECT: COOKING WITH A ROBOT 2 - A VIRTUAL FLOBI COOKBOOK

J. Hemminghaus, T. Schodde, J. Pöppel

Faculty of Technology, Bielefeld University
Bielefeld, Germany

Supervisors: F. Lier, S. Schulz

ABSTRACT

Nowadays, many people are not able to cook for themselves. This paper introduces the visualization, the gesture control and the incorporation of a virtual representation of the anthropomorphic robot head Flobi as part of the Cooking with a Robot (CWAR) project, an interactive system guiding the user step by step through a recipe. These components provide the user with an intuitive interface when trying out new recipes. In order to test the concept as well as the components an experimental study was performed where participants used the system to cook an unknown recipe in an unknown kitchen.

1. INTRODUCTION

Cooking ones own food can be a fun and relaxing activity after an often stressful day as well as save money and improve health. However, in modern society a lot of people do not cook for themselves anymore and mostly have only a very small variety in the recipes they cook. We assume that a big proportion of these people do not cook for themselves because they simply might not know how to cook. Even if these people want to learn cooking, they are often deterred by the complexity of advanced recipes and thus might not have the time or motivation to learn such recipes. In order to encourage less experienced persons to cook more often or to try out new recipes, this system provides them with an interactive kitchen, helping the user during the cooking process. The system uses knowledge about recipes stored in XML files and information about the status of the kitchen hardware, provided by the Miele@Home system. This information is used to guide the user, step by step, through a recipe and inform the user about potential errors, e.g. not preheating the oven. Information about the different steps in a recipe is provided visually in form of a virtual cookbook, which can be projected onto the wall behind the kitchen counter, as well as auditory by reading the steps to the user as well. In order to limit hesitation in using the system, the user interacts with the system using voice and touchless gesture control. Unlike with physical input devices, the user does not need to fear soiling the system when using it after dirtying his or her hands. The voice control allows the user to manipulate the system while moving around freely. Furthermore, we employ a virtual animated representation of the anthropomorphic robot head Flobi [1], providing a concrete entity the user can interact with.

As part of the seminar ”Intelligent Room” at Bielefeld University, three groups have worked on this project. For this purpose, the project has been split into three main parts as can be seen in section 3 System Design. Our group focused on providing an intuitive control method using touchless gestures as well as providing all necessary information to the user in an appealing manner. In this paper, a short overview over existing cooking assistance systems will be given in section 2. After describing the System Design in section 3, section 4 will explain the different components this group developed. In section 5 a video of the system in action is referred to. Section 6 describes the experimental study that was performed to evaluate the systems as well as present the results this group was focusing on. In section 7 the results with respect to this group’s contribution to the project are discussed and possible future improvements to the components regarding the overall cooking system are proposed before finally giving a conclusion in section 8.

2. RELATED WORK

Other research group have already done quite a lot of work regarding systems that assist the user during a cooking process. Systems like PersonalChef [2] and CounterActive [3] use projections and displays in the kitchen in order to provide the user with additional information from a recipe. Using touch input the user can progress step by step through provided recipes. The system eyeCOOK [4] allows the user to control the system using voice commands as well as eyegazes in order to avoid having to touch something while cooking. Another system called Kochbot [5] provides voice control as well and is portable to tablets or smartphones.

While all of these systems provide useful information to the user, there is no publication where status information from the actual kitchen hardware was incorporated into the system, which is something this system does with the Miele@home interface. The Kochbot team did present a newer version of their system at CeBIT this year using the Universal Remote Control technology (OpenURC[1]) to achieve something similar to this approach but they did not publish their new system yet.

3. SYSTEM DESIGN

The interactive kitchen consists of three main parts, distributed to three groups (see Fig. 1): First of all there is the backend, providing the recipe and hardware information as well as managing the actual cooking process. For more information about these components we refer to the report paper of CWAR-1[6]. Another part is the voice control, recognizing user voice commands and providing auditory information to the user. For more information about the dialogue components we refer to the report paper of CWAR-3 [7]. The last part, which is described in further details in this paper (see section 4), provides the visualization of the cookbook, the incorporation of the robot Flobi, as well as the gesture control and synchronization between voice and gesture control for the backend.

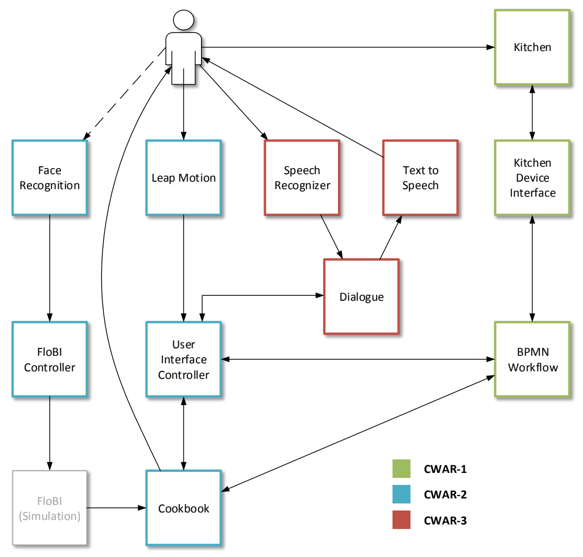


Figure 1: Component diagram of the entire system, showing the distribution of tasks.

[1]see www.openurc.org

4. SOFTWARE COMPONENTS

In this section details about the different software components this group contribute to the overall project are given. These are the actual cookbook visualization (4.1) with its backend (4.2), the user input via gestures (4.3), the face detection (4.5) and the FlobiDriver (4.6).

4.1. The Cookbook (GUI)

The cookbook is a graphical user interface, which displays all necessary and important information about the system. It looks like a real book with a hardcover and pages that can be turned (see Fig. 2). In this way a familiar style for the GUI has been created, which is intuitive for people of all ages.

The cookbook is created with HTML5 and JavaScript in combination with an external framework named TurnJS [8], which mainly provides the ”real book” behaviour but not the style. The communication between the whole system and the GUI uses websockets. To achieve this a server-software named ”pywebsocket” [9] has been used which provides the flow of information between Javascript and the rest of the system. The last used component is the ”MJPEGCANVAS” [10] to display the stream of the virtual agent Flobi [1].

In addition all pages of the book will be added dynamically after getting information from the GUI-backend. The following types of string messages can be send to the JavaScript via the websocket:

RECIPELIST: *recipe_1, recipe_2, . . .*
Add a new page with all recipes which can be cooked with the system. The first recipe will be highlighted as default.

UP/DOWN
Change the highlighting in the recipe list to the next recipe above/below.

SETRECIPE
Return the name of the highlighted recipe in the recipe list via websocket.

RECIPEINIT: *totalAmountOfSteps*
Set the maximum number of steps in the JavaScript in order to know how many pages are required to complete the recipe. Also this information is used to adapt the size of the page stack at both sides of the cookbook, so that it looks more realistic.

INGREDIENTLIST: *recipename; ingredient_1,count_1, unity_1; ingredient_2,count_2,unity_2; . . .*
Add a new page displaying all the required ingredients including their amount for the chosen recipe.

ACTIVSTEP: *stepID; recipename; stepinformation*
Add a new page with information about the next step in the recipe. This page will also contain the name of the recipe and an image of the final food. Afterwards the cookbook will turn automatically.

NEXTPAGE
Turn to the next page.

INFO: *type; message*
Add a new error (red), warning (yellow) or information (blue) message to the current left page of the cookbook (see Fig. 2).

DINFO: *type; message*
Add a new error (red), warning (yellow) or information (blue) message to the next left page of the cookbook. It is needed if a message wants to be displayed after turning the page. This functionality was necessary in the implementation because the model of CWAR-1 always sends the information about the kitchen first and messages which for example would cause a page turn afterwards.

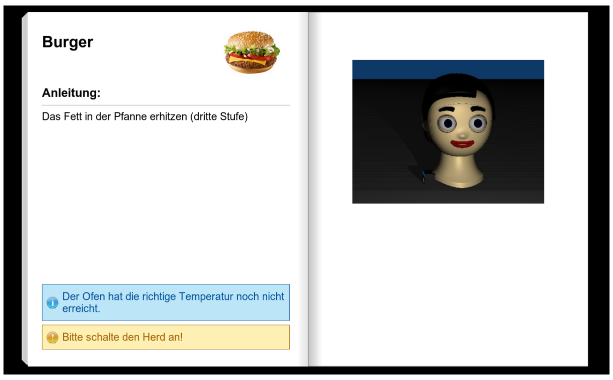
DELINFO
Delete the current warning and information messages.

TIMERINIT: *seconds*
Add a timer-bar at the bottom of the current left page and initialize it with the given time.

TIMERUPDATE: *secondsLeft*
Updates the timer-bar at the bottom of the current left page to the given time.

END
Close the cookbook at the end of the recipe.

There are some different types of sites provided by the cookbook. At first the front- and back-cover which display only Flobi. After the book has been opened, Flobi will be seen on the left and a recipe list on the right. The currently selected recipe is highlighted in green. After starting the cooking process by choosing a recipe, two pages with information will be displayed as can be seen in Fig. 2. All cooking information, like necessary ingredients or current step information will be displayed on the left side of the cookbook. In addition, this site provides warning and information messages about the state of the kitchen hardware regarding the cooking process of the current recipe. On the right side you can see the virtual agent Flobi.


Burger

Anleitung:

Das Fett in der Pfanne erhitzen (dritte Stufe)

Der Ofen hat die richtige Temperatur noch nicht erreicht.

Bitte schalte den Herd an!

Figure 2: Picture of the virtual cookbook - Left: Information about the current step in the cooking process; at the bottom an information (blue) and warning (yellow) message showing the current state of the kitchen. Right: MJPEG-CANVAS with the Flobi-Stream.

4.2. GUI-Backend

As mentioned in section 4.1, a backend for the communication with the GUI has been developed which is written in Python. The main function of the backend is the communication via the Robotic Service Bus (RSB) [11] with the controller and the model from CWAR-1 to provide the GUI with all necessary information. These are redirected to the GUI on a predefined websocket scope for further processing. Additionally the GUI can communicate over the websocket server with the Python class.

4.3. User Input

There are two different possibilities for the user to interact with the developed system. The first type of input allows the user to communicate via German speech which is developed by CWAR-3. The other type enables the user to interact with the cookbook by using hand gestures. For this the Leap Motion [12] is used, which is small enough to be build into the kitchen furniture. The Leap Motion is able to

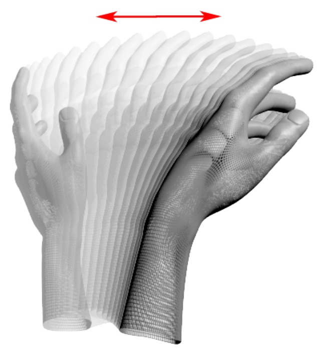


Figure 3: Motion of the swipe gesture [12].

detect hands, fingers and pointing tools via two monochromatic IR cameras and three infrared LEDs in an area of an inverted pyramid centred on the device. Each time the Leap Motion recognizes tools in its view area, it updates frame data which contains information about the detected tool as basic tracking data and motion informations like gestures. The LeapSDK comes with predefined gestures like circular and swipe movements and also a key and a screen tap. For the system the *swipe gesture* is used, a well known gesture for moving to the next page on smartphones or tablets, to turn the pages in the virtual cookbook. Each time the Leap Motion detects a swipe gesture, a notification is send to the listening components via RSB depending on the swipe direction.

4.4. Synchronization of User Input and System Outputs

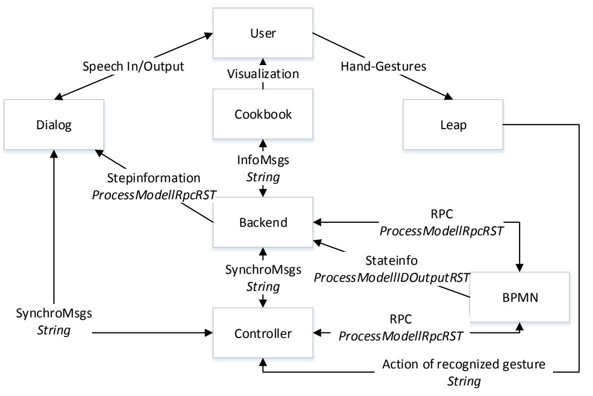


Figure 4: Communication between Controller, Cookbook, Speech and Kitchen Backend.

As mentioned before the system provides two different interaction possibilities, the speech input and the gesture control via Leap Motion, which will be called input devices in the following text passage. In order to prevent the user from accidentally doing multiple inputs to the system, a control unit with different safeguards is developed. If all safeguards are passed the controller handles the message and, depending of the current state in the cooking process, sends it over different RSB Scopes. It can be possible, that the controller receives interaction messages from both input devices simultaneously. In this case the controller handles them with the principle of first in, first out and the other one will be ignored. Additional the control unit is used for the synchronization between the speech output and the visualization in the cookbook, which will be called output devices in the following text passage. Via the controller the two output devices tell each other in which state they currently are.

The figure 4 shows the communication flow between the controller, dialogue system, cookbook and the kitchen backend, which will be illustrated by the following interaction example.

After starting all system components, the cookbook is visualized closed over the kitchen counter. Additional the cookbook backend asks the kitchen backend for the current recipe list and sends it to the cookbook, which stays closed. When the controller receives the first message **”PAGE_FORWARD”** from one of the input devices, it starts the cooking process and sends **”START”** to the output devices. When receiving the message, the Dialogue system starts the welcoming dialogue and the cookbook turns to the first page showing the current recipe list. Then the cookbook backend tells the Dialogue via the controller that the current recipe list is displayed with the message **”GETRECIPELIST”**, which triggers a new dialogue pattern. The user can choose a recipe by saying the recipe name to the Dialogue or using the gesture control to select the highlighted recipe. Both input devices send **”SETRECIPE:recipename”** to the controller, which will redirect this message to the kitchen backend via RPC calls and sets the cooking process to active. Now the kitchen model publishes the current state continuously, which is recognized in the cookbook backend. This enables the backend to ask for the general information of the chosen recipes, like maximum steps, number of ingredients used. But before it asks for a list of ingredients used in the recipe, it sends the controller **”INGREDIENTSLIST”** which is redirected to the Dialogue so it can update its pattern. The next recognized user input starts the recipe and sends to the cookbook backend **”STARTRECIPE”**. Because of this the cookbook backend asks for the first recipe step and directly sends the step information to the Dialogue system for reading it and to the cookbook, which turns to the next page, where the step information is displayed. Now each time the user finishes a step via speech or gesture control the controller notices it and tells the kitchen backend to finish the current active step. Then the kitchen backend will update its status and publishes it to the cookbook backend, which then asks for the new step and sends the received information to the dialogue system. This loop goes on until the kitchen backend tells the cookbook, that the recipe is finished. Then the cookbook sends **”END”** to the controller, which publishes it to the Dialogue.

4.5. Face Detection

The face of the user is detected using haarcascade classifiers with OpenCV [13]. One classifiers is employed to detect frontal faces and two additional classifiers are trained to detect the profiles of the user. The profile classifiers are used in order to improve face detection in cases when the user is not directly looking towards the camera. To improve performance the camera images are rescaled to 320×240 pixel. Concerning the classification parameters, a scale factor of 1.3 was chosen and the minimum number of neighbours was set to 4, since these values proved to yield good results while still ensuring good performance in conjunction with the downscaled images. If one or both profile classifier and the frontal classifier detect a face, then only the detected face of the frontal classifier is used. This is done because the precision of the frontal classifier is higher then the precision of the profile classifiers. Whenever a face is detected, the position and size of the face's bounding box in the original image is published using RSB [11]. These bounding boxes will then be used by the FlobiDriver (see section 4.6) to calculate joint angles for the virtual robot which can be seen in figure 5.

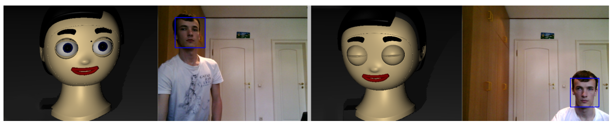


Figure 5: Two snapshots of interaction between face detection and FlobiDriver. The left images show the resulting Flobi looking towards the user. The right images show the camera images with the bounding boxes around the face.

4.6. FlobiDriver

The FlobiDriver calculates the joint angles necessary to make Flobi (in the simulation [1]) look towards the target. In most cases these targets are faces that have been detected by the face detection (see section 4.5), but the targets can also be predefined locations in the kitchen, e.g. the fridge. To compute these angles the relative position of the target with respect to the robot (i.e. the projection) needs to be known. For static targets, such as the fridge this can easily be achieved by measuring the positions of the projection as well as the desired targets in the actual setup. These positions are stored in a config file which is read when the FlobiDriver is started. This approach does not however work for moving faces, for obvious reasons. In order to estimate the 3D position of a detected face the detected bounding box, which is published by the face detection, is used. Using the position in the camera image of the bounding box, as well as information about the camera, which needs to be stored in the config file as well, the direction relative to the camera in which the face was detected is estimated. Initially, a simple mapping of the size of the bounding box to the distance of the face, which also needs to be stored in the config file, is used. The problem with this approach however is, that the size of the bounding box varies very strongly between different users at the same distance. In order to avoid this problem, the mapping between bounding box size and distance would need to be adapted to every user, which was deemed impractical for a kitchen system with potentially an infinite number of different users. In order to still be able to estimate a position, the system simply assumes that the users would always work in a similar distance to the camera and uses this distance for all faces. Once the position of a face is known, the joint angles for the Flobi robot or the simulation can be computed using simple geometry. In this work Flobi always turns his head towards the target, which means that only the pan and tilt angles for the neck need to be computed.

In order to make Flobi look more alive a simple mechanic was implemented which lets him close his eyes in a pseudo-random manner. In each cycle Flobi has a five percent chance to close his eyes. If the eyes are closed, they will stay closed for at least 2 cycles, to avoid extremely rapid blinking. On top of that we also fixed Flobi's mouth into a smiling position. All calculated joint angles are then published to the simulation using ROS [14].

5. INTERACTION EXAMPLE

Interaction with this group's components can be seen in earlier interaction videos[2]. The first video shows the combination of face detection and FlobiDriver. In the second video the gesture control is used to turn the pages of the cookbook. A video showing the entire system at work will soon be available on the course website[3]. In that video extracts from the experimental study will be presented and the contribution of all the three groups will be highlighted.

6. EVALUATION

This section provides information about the experimental study the three CWAR groups performed together, as well as the results relevant to this paper. These will then be discussed in section 7.

6.1. Study design

In order to evaluate the effectiveness and usability of the entire kitchen system, the three CWAR groups designed and performed a small experimental study using the system in the CITEC kitchen at Bielefeld University. For this a beamer, a webcam, the Leap and some speakers were installed in the kitchen. Furthermore, a wireless microphone, that participants were equipped with, was used for the dialog components. For this study eleven participants (8 male, 3 female with an average age of 27 years), who had no contact with the system prior to the study and had varying

[2]see `http://www.techfak.uni-bielefeld.de/isy-praktikum/WS13SS14/CWAR-2/media/IsyFloBi.mp4` and `http://www.techfak.uni-bielefeld.de/isy-praktikum/WS13SS14/CWAR-2/media/Leap_Gesture.mp4`

[3]see `http://www.techfak.uni-bielefeld.de/isy-praktikum/WS13SS14/`

cooking skills, were asked to cook a hamburger with the help of the interactive kitchen system. All participants were given a brief introduction about the kitchen hardware and how to start the system. Concerning the possible controls, the participants were only told, that they can use speech and swiping gestures to control the cookbook. All further instructions during the cooking process were provided by the system itself. Prior and after the actual cooking process, the participants answered two questionnaires. The first one asked general questions about the participants and their cooking expertise. In the second questionnaire the participants were asked to rate the usability of the overall system as well as specialized questions concerning the different groups. This paper focuses on usability of the entire system and, more specifically, of the gesture control and the visualization, as well as the perception of the robot Flobi.

6.2. Results

The results of the questionnaire that this paper focuses on can be categorized into four blocks: Overall usability of the entire system, perception of the robot Flobi, visual attractiveness of the cookbook and usability of the gesture control with the Leap.

For the first block the system usability scale [15] was used and the system achieved a score of 49.3% which indicates severe usability problems.

The results for the perception of Flobi are summarized in table 1.

Perception of Flobi	Average Score
Attentive	2.18
Friendly	1.82
Helpful	2.36
Annoying	3.91

Table 1: Average scores of the perception of the virtual representation of Flobi. Scores ranged from 1: Agree completely to 5: Disagree completely.

The participants rated the cookbook with an average score of 2.64 on a scale from 1: Very appealing over 3: Neutral to 5: Not appealing at all.

The usability of the gesture controls using the Leap was rated with an average score of 3.45 on a scale from 1: Very intuitive over 3: Neutral to 5: Very cumbersome.

Concerning the individual comments regarding the system and cooking experience it can furthermore be report that most participants see a lot of potential in the system, which is so far hindered by various control problems.

7. DISCUSSION

In the following the results from the experimental study will be discussed with respect to the components described in this paper as well as possible future improvements and features.

The rather poor SUS score indicates severe usability problems of the system. The most important reason for this are the user inputs, which often were not working as good as intended. The below average score for the usability of the gesture control using the Leap indicates this as well. Although the gesture control was not used too much by the participants, some users had to repeat the swipe gesture several times before it was recognized. A reason for this poor performance could be the provided framework from the LeapSDK, which is used to detect the gestures. A newer version of this framework has been published after the study was finished. New tests are now needed to evaluate whether the newer version improves the detection rate of the swipe gesture or if a custom gesture detection would need to be developed in future work. Furthermore, we wanted to evaluate how intuitive the provided controls are for the user in the study. For that reason, we only gave the participants a minimal amount of instructions. With hindsight however, it might have been better to have the participants undergo a short training period, in which they would have had the opportunity to get used to the Leap since the success rate improved drastically once the users got used to how they should perform the swipe gesture. Furthermore, the study indicates that some sort of visual feedback when the user's hand is detected by the Leap could be helpful, especially for users new to the system. A simple symbol in the virtual cookbook which changes its colour when a hand is detected could be added as future work.

On top of that, the study made it very clear that the system needs to enable the user to go back to look at previous steps in the recipe. This is another feature that should be implemented in the future. If the user would have the option to go back to the step the user was before to the misclassified input occurred, the error would not be as severe as it is now. Another future feature is a chat below the image of Flobi displaying what the speech recognition understood as well as what the dialogue system says. This could prove quite useful in understanding erroneous behaviour by the system.

Apart from these possible adaptations, the cookbook was rated above average by the participants, which supports our design. Further visual improvements, such as a leather book cover or adjustable font size could be considered for future work.

The results concerning Flobi are mainly positive as well. Most participants rated Flobi attentive which indicates that the face detection and the FlobiDriver performed well. A concern, that a constant focus on the user by Flobi might be annoying, appears to be unjustified. It is also interesting to note, that Flobi was rated slightly helpful despite only looking at the user. This could mean, that the assumption, that users prefer to have some entity they can identify the system with, is correct. It is surprising, that a smile and a pseudo-random blinking behaviour appear to be sufficient in order to create a friendly appearance. Despite the good results in the experimental study there are several improvements to the face detection and the FlobiDriver that could be implemented in the future. First of all, a more natural transitions between joint states would be desirable as in this work we simply jump to target joint states, which can be strange when Flobi performs larger movements. Ideally, mouth movement which is synchronized to the speech should be employed as well, but this would go far beyond the scope of this course. However, a controller for natural transitions between joint states is currently being worked on by one of the supervisors and could potentially be released in the near future. Another possible future improvement could entail the use of several instead of only one camera in the kitchen. Even with detecting profiles as well as frontal faces, the face of a user can still be lost quite easily when the user is moving in the kitchen. This can also lead to bigger movements of Flobi when detecting a face again after it has been lost. Using cameras from multiple angles would allow to track the face of the user more reliably, which could reduce the number of times Flobi needs to perform big head movements.

8. CONCLUSION

This project wanted to implement the concept of an interactive kitchen, which would help users during a cooking process. This group successfully created an appealing cookbook with an integrated animated representation of the anthropomorphic robot head Flobi, that attends to the user, as well as a touchless gesture control for the cookbook. Furthermore, in synchronized the dataflow between the three groups, especially between the two forms of user input and the backend from CWAR-1. Although the usability of the system proved to be not too good, this can mostly be tracked back to erroneous user input components which need to be improved in the future. Overall our study indicates that the concept and its implementation have got a lot of potential which should be pursued further.

9. ACKNOWLEDGEMENT

We would like to thank all the supervisors that helped us during the past semesters, especially we want to say a special thank you to Andreas Kipp for his invaluable help with the study and Marian Pohling with his help regarding the Miele@Home gateway. Furthermore, we would like to thank all the participants of the study, who greatly helped us identify problems of our system.

10. REFERENCES

[1] F. Lier, I. Lütkebohle, and S. Wachsmuth, “Towards automated execution and evaluation of simulated prototype hri experiments,” in *Proceedings of the 2014 ACM/IEEE International Conference on Human-robot Interaction*, ser. HRI ’14. New York, NY, USA: ACM, 2014, pp. 230–231. [Online]. Available: http://doi.acm.org/10.1145/2559636.2559841

[2] S. Mennicken, T. Karrer, P. Russell, and J. Borchers, “First-person cooking: a dual-perspective interactive kitchen counter,” *Proceedings of the 28th of the international conference extended abstracts on Human factors in computing systems - CHI EA ’10*, 2010. [Online]. Available: http://dx.doi.org/10.1145/1753846.1753992

[3] W. Ju, R. Hurwitz, T. Judd, and B. Lee, “Counteractive,” *CHI ’01 extended abstracts on Human factors in computing systems - CHI ’01*, 2001. [Online]. Available: http://dx.doi.org/10.1145/634067.634227

[4] J. S. Bradbury, J. S. Shell, and C. B. Knowles, “Hands on cooking,” *CHI ’03 extended abstracts on Human factors in computing systems - CHI ’03*, 2003. [Online]. Available: http://dx.doi.org/10.1145/765891.766113

[5] U. Schäfer, F. Arnold, S. Ostermann, and S. Reifers, “Ingredients and recipe for a robust mobile speech-enabled cooking assistant for german,” in *KI 2013: Advances in Artificial Intelligence*, ser. Lecture Notes in Computer Science, I. J. Timm and M. Thimm, Eds. Springer Berlin Heidelberg, 2013, vol. 8077, pp. 212–223. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-40942-4_19

[6] A.-V. Dreyer, T. Schürmann, and H. Kaufhold, “Cooking with a robot - the cognitive kitchen-net,” in *ISY Proceedings Summer Term 2014*. CITEC, 2014.

[7] M. Chromik, P. Pekala, and A. Vorwerg, “Cooking with a robot: The dialogue system,” in *ISY Proceedings Summer Term 2014*. CITEC, 2014.

[8] E. García. (2014) Turnjs. [Online]. Available: http://www.turnjs.com/

[9] G. Inc. (2014) pywebsocketserver. [Online]. Available: https://code.google.com/p/pywebsocket/

[10] R. Toris. (2014) Mjpegcanvas. [Online]. Available: http://wiki.ros.org/mjpegcanvasjs

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

[12] I. L. Motion. (2014) Leapmotion. [Online]. Available: https://www.leapmotion.com/

[13] R. Laganière, *OpenCV 2 computer vision application programming cookbook*. Packt Publishing, 2011.

[14] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2, 2009.

[15] J. Brooke, "Sus-a quick and dirty usability scale," *Usability evaluation in industry*, vol. 189, p. 194, 1996.

COOKING WITH A ROBOT 3 - THE DIALOGUE SYSTEM

Monika Chromik, Patryk Pekala, Alexander Vorwerg

Faculty of Technology, Bielefeld University
Bielefeld, Germany

Supervisors: Anja Philippsen, Birte Carlmeyer, Andreas Kipp, Britta Wrede

ABSTRACT

This project is part of a kitchen assistance program providing different user interfaces. We aim for a speech interface, that that can be accessed via natural sentences and is intuitive in a way, that the user does not need any instruction. We also wanted it to be accessible from everywhere in the kitchen and hands free.
In general we managed to create dialogue system, that works as a full interface to kitchen assistance meeting our usage limitations. In our study the users liked the idea, but had some problems because the speech recognition did not always work with the necessary reliability. We also discuss possible ways to fix this problems in the future.

1. INTRODUCTION

Cooking your own meals can be a fun and relaxing activity during the usually stressed day, as well as a way to save money and improve ones health. However, in our modern society a lot of people do not cook for themselves any more or have only a very small variety in the recipes they cook. We assume that a large proportion of these people do not cook by themselves because they simply might not know how to cook. Even if these people want to learn cooking, they are often deterred by the complexity of more advanced recipes and thus might not have the time or motivation to try and learn such recipes. In order to encourage less experienced cooks to cook more often or try out new recipes, we want to provide them with an interactive kitchen, helping the user during the cooking process. The system uses knowledge about known recipes and information about the status of the kitchen hardware, provided by modern kitchen devices, to guide the user, step by step, through a recipe and informs the user about potential errors, e.g. not preheating the oven. Information about the different steps in a recipe are provided visually in form of a virtual cookbook, which can be projected onto the wall behind the kitchen counter, as well auditory by reading the steps to the user as well. In order to limit hesitation in using the system, the user interacts with the system using voice and touch less gesture control. Unlike with physical input devices, the user does not need to fear soiling the system, when using with dirty hands due to the cooking process and also keeping the food hygienic. The voice control furthermore allows the user to manipulate the system while moving around the kitchen freely. We also employ a virtual animated representation of the robot Flobi, providing a specific entity the user can interact with.

As part of the seminar ”Intelligent Room” at Bielefeld University, three groups have worked at this project. Within this framework the project has been split into three main parts which can be seen in section 3. Our group focused on the dialogue system which provides a verbal interaction interface with the kitchen as well as auditory information about the recipe the user wants to cook. After detailing the system design in section 3, section 4 will describe the different components we developed. In section 5 we describe an interaction example, in which a basic use case can be seen. Section 6 describes the experimental study we performed to evaluate the systems. In that section we also present the results our group was focusing on. In section 7 we discuss the results and propose possible future improvements to our components regarding the overall cooking system before finally giving a conclusion in section 8.

2. RELATED WORK

There have been other approaches to implement a dialogue system which works as a cooking assistant. One approach came from Schäfer et al. in 2013 named Kochbot [7]. Kochbot is a cooking assistant application for smartphones and tablet devices which processes speech input/output and supports German recipes. Like our system, it is able to read out cooking instructions step by step and answer questions during cooking. Furthermore, it is able to perform queries in a large recipe collection, e.g. by searching the collection for certain ingredients, the user wants to cook with. With Kochbot, Schäfer et al. wanted to investigate the use of speech assistance in a task oriented scenario, as well as rapid domain adaptation by utilizing natural language processing techniques. It also provides a graphical user interface, which can be controlled through a touch screen. The speech input has two modes. Standard speech recognition relies on certain keywords after which the user can ask arbitrary questions. The Continuous speech recognition only listens to a few commands which are important for the cooking procedure itself (e.g. next step).

Another approach was Cooking Navi, introduced by Hamada et al. in 2005 [2]. Cooking Navi was designed as a cooking navigation system, to help even novice users to cook recipes without failure, while improving their skills. It optimizes the cooking procedure by providing appropriate instructions to the user at the right time. To support the user, there are textual and auditory instructions, as well as videos showing the given tasks performed by somebody else. That approach is an example of a system where visual and auditory interaction accompany each other and enhance the cooking experience to a great extent. In contrast to our system, Cooking Navi does not allow speech input, but circumvents the problem of dirty and/or wet hands by allowing touch input with a water proof pen.

The approach of Martins et al. in 2008 [5] went another way than the works stated above: Instead of waiting for user input, the system instructs the user constantly. The system was designed to assist the user in several domains, other than cooking, where the goal is not to replace the user but providing assistance while the user performs a procedural task. Like the application introduced by Hamada et al. [2], it is also able to play videos to illustrate instructions. The speech input in this system is very rudimentary. Martins et al. allowed a few keywords which can control the dialogue flow, such as 'next' or 'previous' and can get detailed instructions by asking 'how'.

Our approach is very similar to the Kochbot application introduced in [7]. Other than Kochbot, our system does not provide touch support but gesture control (see CWAR-2 [3]), because that way the user does not constantly need to keep his or her hands clean to interact with a touch screen, but he or she can just perform a gesture in which there is no physical touch with potentially delicate or unhygienic hardware. Furthermore, our system is not just an app that runs on a mobile device, but is connected to the kitchen hardware, giving more support for the whole cooking procedure. We also implemented a continuous speech recognition which does not only listen to certain keywords but allows natural sentences due to a large grammar to provide a natural speech experience.
The design of our system in conjunction with the systems of the other two groups ([1] and [3]) is described in the following section.

3. SYSTEM DESIGN

The interactive kitchen consists of three main parts (see figure 1): First of all there is the backend, providing the recipe and hardware information, as well as managing the actual cooking process. For more information about these components we refer to the report paper of CWAR-1[1]. Another part is the visualization, which contains the visual representation of the cookbook, the incorporation of the robot Flobi, as well as the gesture control and junction between voice and gesture control for the backend. For more information we refer to the report paper of CWAR-2 [3]. The last part, which is described in further detail in this paper (see section 4), provides the dialogue system which provides the user with auditory feedback about the cooking process (i.e. the current step in the recipe) as well as a verbal interface to interact with the kitchen.

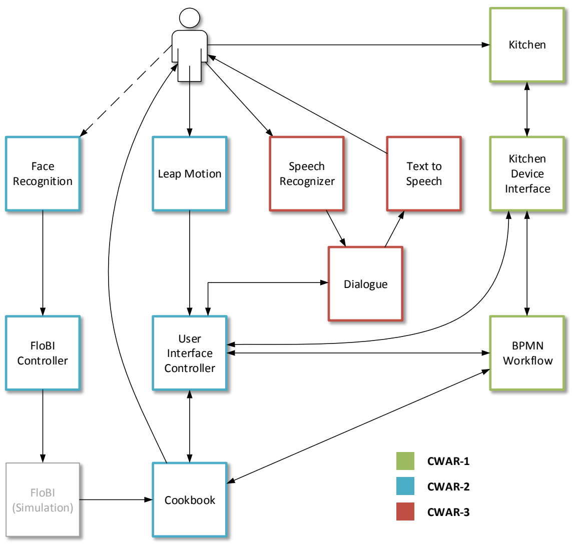


Figure 1: Overview of the entire CWAR project. Our group focussed on the dialogue system, the speech recognizer and the text to speech environment

4. SOFTWARE COMPONENTS

The entire kitchen system consists of many software components which work together on different levels, which can be seen in figure 1. The focus of this paper lies on the dialogue system.
The dialogue system itself is subdivided in three subsystems, which have different functions. The main part consists of the PaMini [6] dialogue framework which provides state machine like interaction patterns for different possible dialogue scenarios. The speech recognition is provided

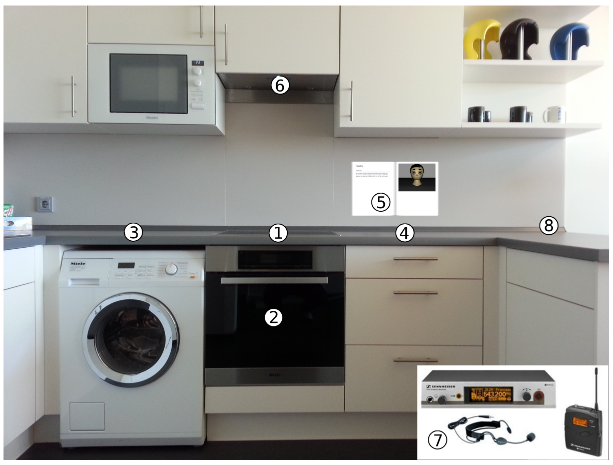


Figure 2: Picture of the kitchen setup
1: stove, 2: oven, 3: work surface, 4: Leap Motion Controller, 5: virtual cookbook, 6: cooker hood, 7: wireless headset, 8: loudspeaker

by PocketSphinx [4]. The speech output is provided by the Mary Text-To-Speech environment [8]. All those components are connected by our state listener class via RSB [9] , which also handles incoming events from the process model. The structure of the dialogue system and a basic dialogue flow is shown in figure 3.
The following sections contain basic descriptions of the components named above. A detailed description of the dialogue flow and the interface with the process model can be found in section 4.4.

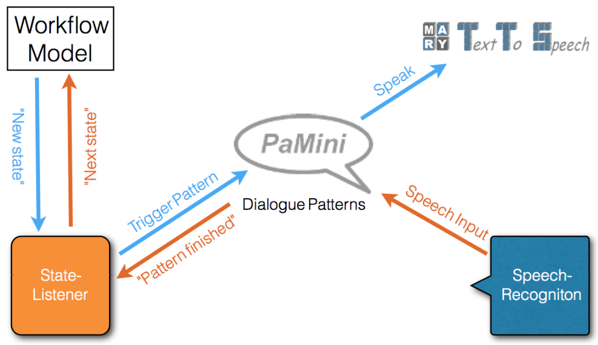


Figure 3: Internal structure of the dialogue system. The heart piece are the PaMini dialogue patterns, handling the PocketSphinx speech recognition and the changing the internal states caught by the state listener. The state listener then triggers new dialogue patterns and requests information by the operating system, corresponding to the finished patterns.

4.1. PaMini dialogue patterns

PaMini (**Pa**ttern-based **Mi**xed-**ini**tiative human-robot interaction) [6] is a framework, which provides interaction patterns for verbal human-robot interaction. With these interaction patterns, more complex dialogues can be realised and more situations can be covered than by providing simple keyword-spotting or command-control techniques.
By providing dialogue patterns, PaMini can handle mixed-initiative situated interaction, which is crucial for the cooking assistant. The dialogue patterns represent reoccurring conversational structures for different situations. The complexity of these patterns go from simple notifications to action requests, that have to be confirmed and/or are cancelable. Each pattern can be understood as a state machine, which covers the current state of the conversation. Every pattern has an 'initiated' and a 'finished' state. Depending on the pattern it can also have several intermediate states, which can be reached depending on the input from robot and/or human. An example of such a pattern is given in figure 4.
The patterns provided by PaMini can be understood as classes of patterns. Systems that use the PaMini framework can build instances of these patterns, which can be configured and customized independently. That way a simple-statement pattern, for example can be customized for different situations. In one situation the human has finished a cooking step and wants to know the next one. In the other situation the human can tell the robot, that he wants the last step to be read out again. Depending on the human's speech input, different patterns of the same class are triggered.
The configuration is done via an XML-document which stores all the information, that is needed for the pattern, i.e. when to go to the next state, which input is accepted and what output will be generated.

Every dialogue should at least contain an opening and a closing pattern. For our implementation, we also used the patterns **HumanSimpleStatement** (i.e. for getting the next recipe step), **RobotSimpleInformationRequest** (i.e. for selecting a recipe), **RobotSuggestion** (i.e. for collecting ingredients) and **RobotSimpleStatement** (i.e. for letting the user know when the cooking starts).

4.2. Speech recognizer PocketSphinx

The speech recognizer PocketSphinx [4] is part of Carnegie Mellon University's speech recognition systems CMU Sphinx. In this project we used PocketSphinx as a real-time continuous speech recognition, with a German language model and middle ware from the Bielefeld University and a grammar we specially designed for our problem.
Our idea for the grammar was to be intuitive but limited to

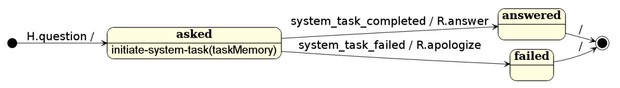

H.question /
asked
initiate-system-task(taskMemory)
system_task_completed / R.answer
answered
system_task_failed / R.apologize
failed
/
/

Figure 4: The HumanInformationRequest pattern with the ’initiated’ and ’finished’ states. A question asked by the human triggers the ’asked’ state and a system task which is responsible for gathering the desired information. When the information can be provided, the state ’answered’ is triggered, otherwise the ’failed’ state is triggered. Both states trigger the ’finished’ state.

our domain, so that the user can talk in normal sentences, without learning keywords and that way being able to use the system without any instruction or manual. But besides that, our grammar only consist of sentences that are expected in the context of cooking and using our program, including questions about the use of the kitchen, cooking devices or cooking procedures. In that way limiting the sentences that the speech recognition understands to those likely to be used and avoiding misunderstanding.
The recognized sentences then are mapped to meanings, that can be processed by the dialogue patterns. These meanings are then forwarded to the dialogue controller, to resume in the activated pattern or start a new one.
We decided to stop speech recognition while speech output occurs (see next section), so that nothing said by our system will be interpreted as speech input.

4.3. Mary Text-To-Speech Environment

For the voice output we decided to use the MARY Text-to-Speech System [8] with a local Mary server for the speech synthesis. We decided to use a local server, so that we would not rely on external sources and connection to the web and still the occupation of memory and computation kept in reasonable bounds. The sentences that arrive from the dialogue controller as an output, are first processed for optimal voice output. For example umlauts cannot be processed by the whole system, so their substitutes have to be replaced with umlauts for the synthesis. These modified sentences are then sent to the Mary server for output.
In our configuration Mary receives plain text as input and speaks in a German female voice with hidden semi Markov model. While speaking, the output is locked, preventing two or more outputs from being spoken at the same time. Outputs that are being blocked this way, are queued and spoken one after an other.

4.4. StateListener and Dialogue Flow

The dialogue flow and the interconnection of the components can be viewed in figure 3. The state listener class fulfils several tasks, that are important for both the user interaction and the interaction with the process model. First of all it handles new incoming events, which represent either new states in the process model or certain situations in the cooking procedure. The events are not received from the process model itself, but from an external class which is responsible for fetching new process model data for each step. The class has been implemented by the CWAR-2 group [3].
Second, it handles events, that occur when dialogue patterns change their state. To accomplish this task, a pattern reporter class has been implemented, which reports to the state listener class.
At last, it also fulfils the task to report to the process model whether a cooking state has been completed. To accomplish this task, the state listener class reports to an external decision maker class which handles input from both the Leap Motion control and the dialogue system. This also has been implemented by the CWAR-2 group.

The pattern reporter class mentioned above, is a class that handles occurring pattern events. These pattern events are triggered, when a dialogue pattern is initiated, finished or changes state. Depending on the patterns these events are handled differently. For example, when the user initiates the dialogue with a greeting and the pattern finishes, the pattern reporter publishes an event, which the state listener will handle. The state listener is aware of which pattern has just finished and takes care and communicates with the decision maker, so that the next step can be finished.

All the components stated above are connected via RSB. Each component listens to or publishes events on its own scope. The state listener class listens to many different scopes and handles incoming events. One scope is for handling different cooking situations like selecting a recipe or checking the needed ingredients. Another scope is for handling events that occur when patterns change their state. This can happen when the user communicates, that he finished a cooking step and wants the next one to be displayed. The last scope handles new incoming recipe steps.
The state listener class itself publishes data on different scopes too. One scope is designated to trigger certain dialogue patterns depending on the situation. On the other scope the state listener communicates with the decision maker class stated above and triggers new cooking states.

With all the necessary components connected to each other, a typical dialogue flow has been realised as follows and can be looked up in figure 3: When a new recipe step arrives, it will be published by an external class to the corresponding scope. Since the state listener handles events on this scope, it handles the incoming event. The incoming data is decomposed and the relevant information to be read out is extracted, i.e. the description of the recipe step. With all the necessary meta data a dialogue pattern can be triggered, which reads out the current recipe step to the user. The pattern is triggered by publishing this data to the corresponding RSB scope the PaMini framework listens to. After triggering the pattern, PaMini handles the user input completely on its own, until the pattern has finished. It also handles the speech output, by forwarding the text to be read out to Mary TTS. In the case stated above, the user can acknowledge the current step, which finishes the pattern. As soon as the user is done with the current step, he can trigger a pattern by himself by saying that he is done. As soon as the agent responds to this command, the pattern finishes and the state listener is informed by the pattern reporter class. The state listener handles this event by publishing a 'nextStep' command to the decision maker class' scope. That will get the next step from the process model and a new step can be read out as mentioned before.

To ensure that the next recipe step can be triggered by both the Leap Motion control and the dialogue system, the state listener does not block the interaction, when a pattern is triggered. For example, when the user triggers the next state with a gesture on the Leap, the next step is fetched and published to a scope the state listener listens to. Since dialogue patterns are interleavable and there can be patterns that have not finished, when a new state arrives the patterns are closed to maintain consistence.

5. INTERACTION/OPERATION EXAMPLES

The dialogue system in action can be seen in our earlier interaction video[1]. This video shows a basic dialogue flow where a recipe for lasagne is read out. The interaction can be started by the user by saying 'Hallo Flobi'(Hello Flobi). Flobi asks what the user wants to cook and after that he asks the user whether he or she has all the needed ingredients. The user can reply that he or she has the ingredients or does not. After that, a few steps are read out. After a step has been read out, the user can confirm that he understood the instruction and can proceed to the next step by saying 'fertig'(done). Also a few examples are shown, where the user can ask Flobi about how to use the oven and so on. When all steps are done, Flobi ends the interaction by saying goodbye.

The interaction with the whole system by using dialogue and leap motion control is not much different from using the dialogue alone. The interaction can be started by either performing a swipe gesture or by saying 'Hallo Flobi'(Hello Flobi). After that, Flobi asks the user, what he wants to cook. For our user study mentioned in section 6, we provided the recipe for making a hamburger. A list of needed ingredients is shown to the user via projector and the user can confirm, that he has all the needed ingredients. Flobi then reads out every recipe step. The instruction itself is also projected on a screen by a projector. After the user has completed a step, he or she can proceed to the next step by either saying 'fertig'(done) or performing a swipe gesture. In the meantime, the user can ask questions, i.e. how to use the oven or the hob. As soon as the user has completed all the needed steps, Flobi ends the interaction by saying goodbye and the visual cookbook closes. A video showing the entire system at work will soon be available on the course website[2].

6. EVALUATION

In this section we provide information about the experimental study, the three CWAR groups performed together, as well as about the results relevant to our group. These will then be discussed in section 7.

6.1. Study design

The goal of the evaluation was to find out, whether the cooking system is easy to use and whether the cooking experience is improved by the cooking system. For the evaluation of the cooking system eleven participants were invited. The task for each participant was to cook a hamburger with the cooking system. In order to ensure that all study participants cook under the same conditions, each of them obtained a short introduction of the kitchen hardware like the oven and the induction cooker. The required ingredients were provided for each study participant and the required amount of mincemeat were weight out previously. Also, every participant was instructed that it is possible to both speak and use gestures to interact with the cooking system. The interaction was done in German.
On average the study participants were 27 years old and have had some cooking experience with cooking easy recipes. Ten of them had a lot of experience with computers and five of them with voice input systems. Six out of eleven of the study participants had experience in programming computers and all but two participants, had experience with the robot Flobi or robot systems in general.

6.2. Results

The following evaluation results and analysis refer to the dialogue system. The focus of the evaluation of the dialogue

[1]`http://www.techfak.uni-bielefeld.de/isy-praktikum/WS13SS14/CWAR-3/media/DialogVideo.mp4`

[2]`http://www.techfak.uni-bielefeld.de/isy-praktikum/WS13SS14/`

system was to find out, whether the dialogue system was easy to use and whether the word pool of the voice recognition is diverse enough and how the system reacts to background noises. The idea to use a cooking system with voice input was approved by almost every study participant and was rated by an average score of 2 on a scale from 1, which is very useful, over 3, which is neutral, to 5, which is not useful. They stated that the use of voice is comfortable and easy to use, if you know which sentences the dialogue system understands.
The two most important questions were 'How intuitive is the usage of system by speech?' and 'Did the system not understand often during the cooking process?'. For the first important question the participants could choose between 1, which is very intuitive, and 5, which is not intuitive. In order to figure out whether the dialogue system is easy to use, the study participants were not informed which sentences or keywords they could use, to go on with the dialogue. In this way we were able to find out, which sentences were missing in the word pool of the voice recognition. The result of this experiment was an average score of 2.9. Four of the participants intuitively managed the operation of the dialogue system, four were not sure of how to use the system and three did not understand the system intuitively. The rating of the intuitiveness of the system is shown in figure 5. In most cases the system did not understand the user, which affected the cooking process. Only in four out of eleven cases the system did not understood the user less than four times, which is shown in figure 6. Very interesting is, that although the voice recognition did not work that well, the users have used the speech recognition strikingly more often than the gesture control. If the speech recognition did not work the participant tried to use the gesture control but already on the next cooking instruction he or she again tried to use the speech recognition.

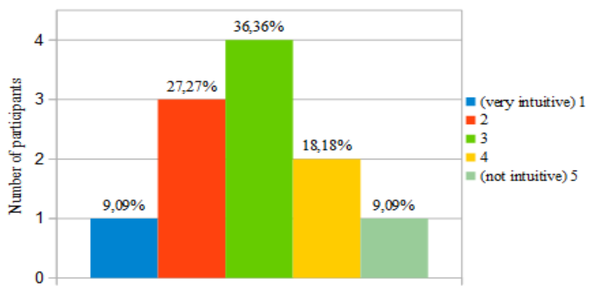


Figure 5: The bar chart shows how intuitive the user perceived the dialogue system. They had the possibility to rate the dialogue system with a number between one and five. One means very intuitive and five means not intuitive.

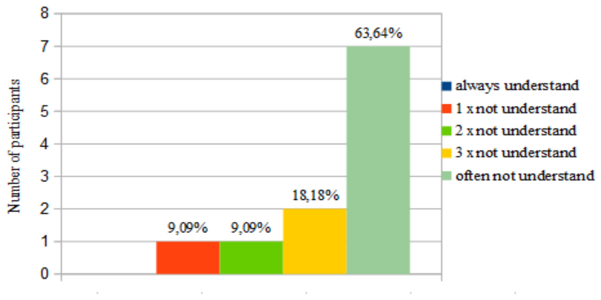


Figure 6: Illustrates how often the dialogue system did not understand the user. It could be chosen between always understand, once, twice, three times and often not understood.

7. DISCUSSION

The reason for this sobering result lies in the grammar of the voice recognition, which is still at a premature stage, and the problem that the voice recognition often misunderstood what the user said. The system repeated the current cooking instruction very often, because a lot of sentences and questions, which the participants said, were not integrated in the grammar. Also a big problem were the background noises, like the running cooker hood and the rustling of plastic bags or onion skin when the participants peeled an onion. The system recognized the noise very often as a sentence, such as “Ich bin fertig”(“I am ready”), which is in the grammar of the voice recognition, and therefore the assistant proceeded to the next step and read the next cooking instruction to the participant, who was not finished with the previous step. Some study participants thought aloud or talked during cooking, which was a further problem for the dialogue system. The system did not understand the user, either it repeated the current cooking step or it went on to the next step. The reason that the system proceeded to the next step so often might be that due to our goal of being as intuitive as possible, there were many possible sentences for important instructions. This lead to a lot of chances where noise or language, not directed to the assistant, was misinterpreted as the instruction to go to the next step.
A possibility to improve the speech recognition could be the use of a keyword like “Flobi” in the beginning of each sentence directed to the assistant. Only if the Keyword “Flobi” is called, the system will listen and does not try to match the noise to some words of the word pool. Another possibility could be leaving the open speech recognition design and instead accept only fixed commands such as “fertig” (“ready”) and “weiter” (“next”). This way the usage of the dialogue system could be easier and more stable but the user has to know which commands to use, to interact with the system or a compromise like only using keywords for interactions that occur very often. These are limited in number

and can that way be learned quickly and reduce the chance of being triggered by noise or talking, which is not directed to the assistant while still containing the opportunity of an open designed grammar for the rest of the interaction. Another possibility could be to use another speech recognizer and a very large word pool and grammar, which covers also statements not related with the assistant in any way, to filter those out. But besides the problems that occurred with many participants, there were also participants who worked quite well with the assistant. Those runs let assume, that our general approach to communicate via language with the cooking assistant, is an appropriate way and it may be possible to keep this system nearly as self explanatory as it is and thus not constructing a threshold to use it.

8. CONCLUSION AND OUTLOOK

In conclusion our approach aims at the right direction, because it showed to be useful to access a cooking assistant via voice control and thus being hands free, hygienic and free to move around in the kitchen. But our goal to a design with intuitive access and natural sentences lead to some problems which decrease the usability of the assistant fundamental.
So in the future, the system can be equipped with different speech recognition designs, comparing a general keyword for sentences directed to the system, keywords for each interaction with the system or an open design and with keywords only for the commands that occur very often to ensure usability, reliability and intuitivity.

9. REFERENCES

[1] Adriana-Victoria Dreyer, Hauke Kaufhold, and Tim Schürmann. Cooking with a robot 1 - the cognitive kitchen net. In *ISY Proceedings Summer Term 2014*. CITEC, 2014.

[2] Reiko Hamada, Jun Okabe, Ichiro Ide, Shin'ichi Satoh, Shuichi Sakai, and Hidehiko Tanaka. Cooking navi: Assistant for daily cooking in kitchen. In *Proceedings of the 13th Annual ACM International Conference on Multimedia*, MULTIMEDIA '05, pages 371–374, New York, NY, USA, 2005. ACM.

[3] Jacqueline Hemminghaus, Thorsten Schodde, and Jan Pöppel. Cooking with a robot 2 - a virtual flobi cookbook. In *ISY Proceedings Summer Term 2014*. CITEC, 2014.

[4] D. Huggins-Daines, M. Kumar, A. Chan, A.W. Black, M. Ravishankar, and A.I. Rudnicky. Pocketsphinx: A free, real-time continuous speech recognition system for hand-held devices. In *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, volume 1, pages I–I, May 2006.

[5] F. Martins, J.P. Pardal, L. Franqueira, P. Arez, and N.J. Mamede. Starting to cook a tutoring dialogue system. In *Spoken Language Technology Workshop, 2008. SLT 2008. IEEE*, pages 145–148, Dec 2008.

[6] Julia Peltason and Britta Wrede. Pamini: A framework for assembling mixed-initiative human-robot interaction from generic interaction patterns. In *Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 229–232. Association for Computational Linguistics, 2010.

[7] Ulrich Schäfer, Frederik Arnold, Simon Ostermann, and Saskia Reifers. Ingredients and recipe for a robust mobile speech-enabled cooking assistant for german. In Ingo J. Timm and Matthias Thimm, editors, *KI 2013: Advances in Artificial Intelligence*, volume 8077 of *Lecture Notes in Artificial Intelligence*, pages 212–223. Springer, 9 2013.

[8] Marc Schröder and Jürgen Trouvain. The german text-to-speech synthesis system mary: A tool for research, development and teaching. *International Journal of Speech Technology*, 6(4):365–377, 2003.

[9] Johannes Wienke and Sebastian Wrede. A middleware for collaborative research in experimental robotics. In *System Integration (SII), 2011 IEEE/SICE International Symposium on*, pages 1183–1190. IEEE, 2011.

EXTRACTION OF RELATIONAL DATA FROM PUBLICATIONS ON PRECLINICAL EXPERIMENTS FOR SPINAL CORD INJURY TREATMENTS[1]

Benjamin Paaßen, Andreas Stöckel, Raphael Dickfelder, Jan Philip Göpfert

Faculty of Technology, Bielefeld University
Bielefeld, Germany

Supervisors: Roman Klinger, Matthias Hartung, Philipp Cimiano

ABSTRACT

Preclinical research in the field of central nervous system trauma advances at a fast pace, currently yielding over 8,000 new publications per year, at an exponentially growing rate. This amount of published information by far exceeds the capacity of individual scientists to read and understand the relevant literature. So far, no clinical trial has led to therapeutic approaches which achieve functional recovery in human patients.

Building upon an existing ontology-based information extraction system we elaborate in this paper on how relational content describing experiments on spinal cord injury treatments can be extracted from natural language text. Using machine learning techniques we achieved a test precision of 0.87 and test recall of 0.86.

1. INTRODUCTION

Injury to the central nervous system of adult mammals typically results in lasting deficits, like permanent motor and sensor impairments, due to a lack of profound neural regeneration. Specifically, patients who have sustained spinal cord injuries (SCI) usually remain partially paralyzed for the rest of their lives. Preclinical research in the field of central nervous system trauma advances at fast pace, currently yielding over 8,000 new publications per year, at an exponentially growing rate, with a total amount of approximately 160,000 PubMed-listed papers today.[2]

However, translational neuroscience faces a strong disproportion between the immense preclinical research effort and the lack of successful clinical trials in SCI therapy: So far, no therapeutic approach has led to functional recovery in human patients [Filli and Schwab, 2012]. As the vast amount of published information by far exceeds the capacity of individual scientists to read and understand the relevant knowledge [Lok, 2010], the selection of promising therapeutic interventions for clinical trials is notoriously based on incomplete information [Prinz et al., 2011, Steward et al., 2012].

Thus, automatic information extraction methods are needed to gather structured, actionable knowledge from large amounts of unstructured text that describes outcomes of preclinical experiments in the SCI domain. Being stored in a database, such knowledge provides a highly valuable resource enabling curators and researchers to objectively assess the prospective success of experimental therapies in humans, and supports the cost-effective execution of meta studies based on all previously published data. First steps towards such a database have already been undertaken by manually extracting the desired information from a limited number of papers [Brazda et al., 2013], which is not feasible on a large scale, though.

In previous work we presented a first prototype of an automated ontology-based information extraction system for the acquisition of structured knowledge about experimental SCI therapies [Paassen et al., 2014]. We showed that it is not sufficient to consider entities independent of each other but that information in the SCI domain is highly structured and can be described in terms of relations. Building upon these results we focus on the task of relation extraction using machine learning techniques. We prodive a cascaded workflow for relation extraction in section 3 and evaluate our approach with respect to test data in the machine learning training workflow as well as with respect to an independent evaluation data set (*cf.* section 4). In section 5 we give suggestions for further work and solutions to problems that arose during development.

2. RELATED WORK

Our work in this paper is an extension of [Paassen et al., 2014]. In that contribution, a hand-tailored algorithm was used for relation extraction. It rigidly assumed a high relevance of the proximity between two tokens in the text and equal importance of tokens taking part in relations. Here we loosen those assumptions and attempt

[1]This report is an extended and adapted version of [Paassen et al., 2014].
[2]As in this query to the database PubMed (link to `http://www.ncbi.nlm.nih.gov/pubmed`), as of April 2014.

to make the approach more generalizable.

We heavily rely upon ontologies in order to find meaningful instances of entities in the domain of spinal cord injury treatments within text. Thus, our approach is an example of an ontology-based information extraction system [Wimalasuriya and Dou, 2010]: Large amounts of unstructured natural language text are processed through a mechanism guided by an ontology, in order to extract predefined types of information. Our long-term goal is to represent all relevant information on SCI treatments in structured form, similar to other automatically populated databases in the biomedical domain, such as STRING-DB for protein-protein interactions [Franceschini et al., 2013], among others.

A strong focus in biomedical information extraction has long been on named entity recognition, for which machine-learning solutions such as conditional random fields [Lafferty et al., 2001] or dictionary-based systems [Schuemie et al., 2007, Hanisch et al., 2005, Hakenberg et al., 2011] are available which tackle the respective problem with decent performance and for specific entity classes such as organisms [Pafilis et al., 2013] or symptoms [Savova et al., 2010, Jimeno et al., 2008]. A detailed overview on named entity recognition, covering other domains as well, can be found in [Nadeau and Sekine, 2007].

The use case described in this paper, however, involves a highly relational problem structure in the sense that individual facts or relations have to be aggregated in order to yield accurate, holistic domain knowledge, which corresponds most closely to the problem structure encountered in event extraction, as triggered by the ACE program [Doddington et al., 2004, Ji and Grishman, 2008, Strassel et al., 2008], and the BioNLP shared task series [Nedellec et al., 2013, Tsujii et al., 2011, Tsujii, 2009]. General semantic search engines in the biomedical domain mainly focus on isolated entities. Relations are typically only taken into account by co-occurrence on abstract or sentence level. Examples for such search engines include GoPubMed [Doms and Schroeder, 2005], SCAIView [Hofmann-Apitius et al., 2008], and GeneView [Thomas et al., 2012].

With respect to the extraction methodology, our work is similar to [Saggion et al., 2007] and [Buitelaar et al., 2008], in that a combination of gazetteers and extraction rules is derived from the underlying ontology, in order to adapt the workflow to the domain of interest. The relation extraction approach using binary classifiers bears some resemblance to the best-performing system in the BioNLP shared task in 2011 [Björne and Salakoski, 2011].

A schema in terms of a reporting standard has recently been proposed by the MIASCI (Minimum Information About a Spinal Cord Injury Experiment)-consortium [Lemmon et al., 2014]. To the best of our knowledge, our work (including [Paassen et al., 2014]) is the first attempt at automated information extraction in the SCI domain.

Relation	Slot Entity Classes
Animal	**Organism / Laboratory Animal**
	Gender
	Age / Exact Age
	Weight
	Number
Injury	**Injury Type**
	Injury Device
	Injury Height / Vertebral Position
Treatment	**Drug**
	Delivery
	Dosage
Result	**Investigation Method**
	Significance
	Trend
	p Value

Table 2: The templates for all four considered relations: A slot of a given relation can - in principle - be filled by any instance of a suitable class. Slots in **boldface** mark the *trigger* slot of that particular relation: An instance for that slot is required to trigger the relation extraction mechanism, while all other slots may remain empty.

3. METHOD AND ARCHITECTURE

An illustration of the proposed workflow is shown in Figure 1. Based on the unstructured information management architecture (UIMA, [Ferrucci and Lally, 2004]), full text PDF documents serve as input to the workflow. Plain text and structural information are extracted from these documents using Apache PDFBox[3].

The workflow then performs named entity recognition (NER) (*cf.* Section 3.1). The entity classes extracted from the text are shown in Table 1. Some of the entity classes can be handled by simply using regular expressions or regex-based recombinations of other entities. Many other classes require domain-specific ontologies. Some of those had to be handcrafted for this application and will be released - like the system itself - at `http://opensource.cit-ec.de/projects/scie`.

The proposed system extracts *relations* which we define by templates that contain slots, each of which is to be filled by an instance of a particular entity class (*cf.* Table 2). We argue that a relational approach is essential to information extraction in the SCI domain as (i) many instances of entity classes found in the text do not convey relevant information

[3]Apache PDFBox – A Java PDF Library `http://pdfbox.apache.org/`

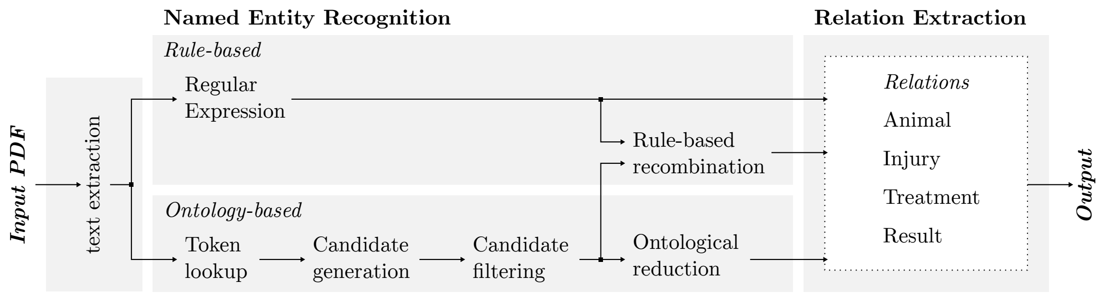


Figure 1: Workflow of our implementation, from the input PDF document to the generation of the output relations. Named entity recognition is described in Section 3.1, relation extraction in Section 3.2.

Entity Class	Example	Method	Resource	Count
Integer	"42", "2k", "1,000"	R	Regular Expressions	
Float	"4.23", "$8.12 \cdot 10^{-8}$"	R	Regular Expressions	
Roman Number	"XII", "MCLXII"	R	Regular Expressions	
Word Number	"seventy-six"	O	*Word Number List*	99
Range	"2-4"	R	QTY + PARTICLE + QTY	
Language Quantifier	"many", "all"	O	*Quantifier List*	11
Time	"2 h", "14 weeks"	R	QTY + TIME UNIT	
Duration	"for 2h"	R	PARTICLE + TIME	
Organism	"dog", "rat", "mice"	O	NCBI Taxonomy	67657
Laboratory Animal	"Long-Evans rats"	O	*Special Laboratory Animals*	5
Gender	"male", "female"	O	*Gender List*	2
Exact Age	"14 weeks old"	R	TIME + AGE PARTICLE	
Age	"adult", "juvenile"	O	*Age Expressions*	2
Weight	"200 g"	R	QTY + WEIGHT UNIT	
Number	"44", "seventy-six"	R	QTY	
Injury Type	"compression"	O	*Injury Type List*	7
Injury Device	"NYU Impactor"	O	*Injury Device List*	21
Vertebral Position	"T4", "T8-9"	R	Regular Expressions	
Injury Height	"cervical", "thoracic"	O	*Injury Height Expressions*	4
Drug	"EPO", "inosine"	O	MeSH	14000
Delivery	"subcutaneous", "i.v."	O	*Delivery Dictionary*	34
Dosage	"14 ml/kg"	R	QTY + UNIT	
Investigation Method	"walking analysis"	O	*Method List*	117
Significance	"significant"	O	*Significance Quantifiers*	2
Trend	"decreased", "improved"	O	*Trend Dictionary*	4
p Value	"$p < 0.05$"	R	P + QTY	4

Table 1: A detailed list of entity classes which we extract from the text, together with examples for instances of each class, the extraction method, and resources used for extraction. "Method" refers to the entity extraction method which is based either on regular expressions (R) or on a lookup in our ontology database (O). Resources in *italics* are ontologies, which were specifically created for this application; resources in SMALL CAPITALS are regular expression-based recombinations of other entities. The count specifies the number of elements in the respective resource.

on their own, but only in combination with other instances (*e. g.*, surgical devices mentioned in the text are only relevant if used to inflict a spincal cord injury to the animals in an experimental group), and (ii) a holistic picture of a preclinical experiment can only be captured by aggregating several relations (*e. g.*, a certain p value being mentioned in the text implies a particular treatment of one group of animals to be significantly different from another treatment of a control group).

We take four relations (*Animal*, *Injury*, *Treatment* and *Result*) into account which capture the semantic essence of a preclinical experiment: Laboratory animals are injured, then treated and the effect of the treatment is measured. Table 2 provides an overview of all relation templates. The relation extraction mechanism is based on machine learning techniques described in Section 3.2.

3.1. Ontology-based Named Entity Recognition

We store ontological information in a relational database as a set of directed graphs, accompanied by a dictionary for efficient token lookup. Each entity is stored with possible linguistic surface forms (*e. g.*, "Wistar rats" as a surface form of the *Wistar rat* entity from the class *Laboratory Animal*). Each surface form $\mathbf{s}$ is tokenized (on white space and non-alphanumeric symbols, including transformation to lowercase, *e. g.*, leading to tokens "wistar" and "rats") and normalized (stemming, removal of special characters and stop words) resulting in a set of *dictionary keys* (*e. g.*, "wistar" and "rat"). The resources used as content for the ontology are shown in Table 1. We use specifically crafted resources for our use case[4] as well as the NCBI taxonomy[5] and the Medical Subject Headings[6] (MeSH). The process of ontology-based NER consists of (i) *token lookup* in the dictionary, (ii) *candidate generation*, (iii) *probabilistic candidate filtering* and (iv) *ontological reduction* (*cf.* Figure 1).

3.1.1. Token lookup.

For each token t in the document, the corresponding surface form tokens $\mathbf{s}_t$ are retrieved from the database. A *confidence value* p_c based on the Damerau-Levenshtein-Distance without swaps (dld, [Damerau, 1964]) is calculated as

$$p_\text{c}(t, \mathbf{s}_t) := \max\left\{0, 1 - \min_{t' \in \mathbf{s}_t} \frac{\text{dld}(t', t)}{|t'|}\right\}, \qquad (1)$$

where $|t|$ denotes the number of characters in token t. Assuming to find $t =$ "rat" in the text with the according surface form $\mathbf{s}_t = (\text{"wistar"}, \text{"rats"})^\intercal$, $p_\text{c}(t, \mathbf{s}_t) = 1 - \frac{1}{4} = 0.75$. Tokens with $p_\text{c} < 0.5$ are discarded.

3.1.2. Candidate generation.

A candidate $\mathbf{h}$ for matching the surface form tokens $\mathbf{s_h}$ is a list of tokens $(t_1^{\mathbf{h}}, \ldots, t_n^{\mathbf{h}})^\intercal$ from the text. Candidates are constructed using all possible combinations of matching tokens for each surface form token (as retrieved above). To keep this tractable, we restrict the search space to combinations with the proximity $d(t_k^h, t_\ell^h) \leq 9$ for all $t_k^h, t_\ell^h \in \mathbf{h}$, where

$$d(u, v) := N_W(u, v) + 3 \cdot N_S(u, v) + 10 \cdot N_P(u, v) \qquad (2)$$

models the distance between two tokens u and v in the text with N_W, N_S, N_P denoting the number of words, sentences and paragraphs between u and v. In our example, the candidate would be $\mathbf{h} = (\text{"rat"})^\intercal$.

3.1.3. Candidate filtering.

For a candidate $\mathbf{h}$ and the surface form tokens $\mathbf{s_h}$ it refers to, we calculate a total *match probability* p_m, taking into account the distance $d(u, v)$ of all tokens in the candidate, the confidence $p_\text{c}(t', \mathbf{s_h})$ that the token actually belongs to the surface form, and the ratio $\sum_{t' \in \mathbf{h}} |t'| / \sum_{t \in \mathbf{s_h}} |t|$ of the surface form tokens covered by the candidate:

$$p_\text{m}(\mathbf{h}, \mathbf{s_h}) = \max_{t \in \mathbf{h}} \sum_{t' \in \mathbf{h}} p^3_\text{dist}(t, t') \cdot p_\text{c}(t', \mathbf{s_h}) \cdot \text{rel}(t', \mathbf{s_h}) \qquad (3)$$

where

$$\text{rel}(t', \mathbf{s_h}) := \frac{|t'|}{\sum_{t \in \mathbf{s_h}} |t|}$$

and

$$p^\sigma_\text{dist}(u, v) := \exp\left(-\frac{d(u, v)^2}{2\sigma^2}\right) \qquad (4)$$

models the confidence that two tokens u and v belong together given their distance in the text. In our example of the candidate $\mathbf{h} = (\text{"rat"})^\intercal$ with the surface form tokens $\mathbf{s_h} = (\text{"wistar"}, \text{"rats"})^\intercal$ is $p_\text{m}(\mathbf{h}, \mathbf{s_h}) = 1 \cdot 0.75 \cdot \frac{3}{6+4} = 0.225$. Candidates with $p_\text{m} < 0.7$ are discarded.

3.1.4. Ontological reduction.

As the algorithm ignores the hierarchical information provided by the ontologies, we may obtain overlapping matches for ontologically related entities. Therefore, in case of overlapping entities that are related in an "is a" relationship in the ontology, only the more specific one is kept. Assume for instance the candidates "Rattus norvegicus" and "Rattus norvegicus albus", where the latter is more specific and therefore accepted.

[4]Resources built specifically are made publicly available at `http://opensource.cit-ec.de/projects/scie`

[5][Sayers et al., 2012], database limited to vertebrates: `http://www.ncbi.nlm.nih.gov/taxonomy/?term=txid7742[ORGN`

[6][Lipscomb, 2000], excerpt of drugs from Descriptor and Supplemental: `https://www.nlm.nih.gov/mesh/`

3.2. Relation Extraction

We frame *relation extraction* as a template filling task, such that each slot provided by a relation template has to be assigned a filler of the correct entity class or remain empty. Entity classes for the four relations of interest are shown in Table 2, where *trigger* slots are in **boldface**, whereas all other slots are *optional*, *i. e.* they may remain empty.

Assume a relation template $\mathfrak{R}$ and an arbitrary numbering of its slots. We identify the i-th slot with i. Denote the set of instances that could fill the i-th slot with $S_i(\mathfrak{R})$ (including the empty instance ϵ for *optional* slots). This set contains all those instances of suitable classes that were found by NER. Note that we are completely dependent on a high recall during NER, as an instance that was not found during NER is not an element of $S_i(\mathfrak{R})$ and therefore cannot be part of a relation. It directly follows from our definition of a *relation* that the set of possible relations is the cartesian product:

$$S_1(\mathfrak{R}) \times \cdots \times S_n(\mathfrak{R}) \tag{5}$$

where n is the number of slots in the template.

This scales exponentially with respect to the number of slots. Our approach for Relation extraction (RE) therefore attempts to limit the size of each set $S_i(\mathfrak{R})$ as much as possible, using a cascaded approach:

1. The RE process is triggered by an instance h of an entity class that fits the *trigger* slot for the template $\mathfrak{R}$. A binary classifier decides if h should be discarded or if the RE process should be continued using h.
2. For each *optional* slot i, a binary classifier that takes h and i into account, is employed to remove instances from $S_i(\mathfrak{R})$.

   Furthermore, we apply a liberal distance heuristic: Only those instances from $S_i(\mathfrak{R})$ are considered that are at most 3000 characters away from h.

   We call the set of the remaining instances $S_i(h)$.
3. We now explicitly construct the cartesian product

   $$R(h) := \{h\} \times \big(\bigtimes_{j \neq i} S_j(h) \big) \tag{6}$$

   where j is the index of the *trigger* slot.
4. A final classifier gives a confidence that some $r \in R(h)$ is indeed a viable relation. Ranked according to confidence, no more than the best $N = 6$ relation instances are kept and no more than $M = 3$ relation instances that overlap in the text.
5. Finally, we remove all relations $r \in R(h)$ that do not meet sanity check conditions. Currently, we only enforce the constraint that the *Number* instance in an *Animal* relation can not be a negative number and that the *Number* and *Weight* instance may not be the same.

As with NER, our method depends on high recall during the first two steps: If some instance of $S_i(\mathfrak{R})$ is not transferred to $S_i(h)$, it cannot be used in a relation. To achieve that, we trained the respective classifiers using a 5-fold cross validation with 3 repeats each and always kept those parameters that achieved the highest recall. This lead to a test recall higher than 95% in all cases but for the trigger classifier on our training data (*cf.* Section 4).

As features for *trigger* classifiers, we used the *relative begin index* and the *relative end index* of the instance in the text, the *relative length*[7] of the instance, and the value of p_m (*cf.* Equation 3). Furthermore, automatically generated identitiy features for the instance text, the part-of-speech tags[8] of words taking part in that instance, and the ontology ID of that instance. Furthermore, word identity features were used for the words between trigger and slot, the relative distance between them and p^σ_dist with $\sigma = 6$ for the *Animal* and *Treatment* template, $\sigma = 10$ for the *Injury* template and $\sigma = 15$ for the *Result* template. For relation classifiers we used the *relative begin* and *end index* of the whole relation, the *relative size*, the number of text tokens inside the relation as a numerical feature and as several binned, binary features in steps of 5 (e.g. "less than 15 tokens"), word identity features for the tokens the number of filled slots, binary features for each slot if it was filled or not and the confidence of each previous classifier.

To acquiere such confidences we used logistic regression in the implementation provided by the library *liblinear* [Fan et al., 2008] and its implementation by Waldvogel[9]. As the number of features for the classifiers is very high (1988 features on average), a strict regularization is required to avoid overfitting/emphasis on features that are not generally relevant. Using an L1-regularization we were able to effectively select features during training, thereby reducing the number of features to 63 on average (a reduction of about factor 30).

4. EXPERIMENTS

4.1. Classifier Training & Testing

We trained our classifiers on a corpus of 18 complete papers that were annotated in a two-step process: First, NER was applied. Then, all meaningful relation instances that could be formed from the instances extracted by NER, were annotated manually as positive examples (e.g. NER detected "Sprague-Dawley rats" as a *LaboratoryRat* instance and "female" as a gender instance, where "female" indeed describes the gender of the rats mentioned. A human expert then notes that "female Sprague-Dawley rats" is a proper *Animal*

[7]*Relative* meaning divided by the length of the paper text.

[8]POS tagging was done using the Stanford POS Tagger: `http://nlp.stanford.edu/software/tagger.shtml`

[9]`http://liblinear.bwaldvogel.de/`

Relation	Prec.	Recall	F_1
Animal	0.63	0.60	0.59
Injury	0.24	0.40	0.30
Treatment	0.39	0.26	0.31
Result	0.23	0.09	0.13

Table 4: The system's performance in identifying the places in the text that are fillers of a meaningful relation. For each relation the average precision, recall and F_1-measure is given.

relation instance). Negative examples were automatically generated from the remaining possible relations. This lead to an imbalanced data set set (12 (median) times more negative examples than positive ones) which was balanced by random oversampling of the positive examples. As discussed before, training was carried out in a cross-validation setting with 5 folds and 3 repetitions each.

As discussed before, the relation classifiers depend on all previous classifiers. Therefore, we only report the results for the four relation classifiers here. We state the number of data points used, as well as precision, recall, F_1-measure, and the area under the reciever operator curve (ROC), for training as well as for test. The ROC was computed by varying the acceptance threshold of the classifier: E.g. for a threshold of 0.9, only those data points were classified with a positive label, that had a logistic regression confidence larger than 0.9. The threshold was varied between 0 and 1 in steps of 0.02. The results are presented in Table 3. F_1-measure is consistently between 0.85 and 0.90 for all relations in the test setting, while area under curve is as high as between 0.90 and 0.94. For almost all relations, precision and recall are fairly balanced. The only exception is the *Treatment* relation, with 0.79 precision and 0.92 recall. With regard to runtime, we found that about 23 microseconds are needed per data point during training, including all I/O handling and data preprocessing, leading to about an hour of training time on a four year old AMD Quad Core machine.

4.2. Independent Evaluation

To analyse improvement over the baseline set in [Paassen et al., 2014] we repeated the experiment done there: We used the whole pipeline on a set of 32 papers not previously used during training or development. The 32 papers were annotated by medical experts only with respect to instances of entity classes that are fillers in a relation (1186 instances in total). We compared these expert annotations with the ones done by our system (998 in total) and denoted a true positive only if the expert annotation and the system annotation refer to the same position in the text (in [Paassen et al., 2014] this is called the *annotation* task in the *fillers only* setting).

We report the average results for each relation in table 4. For the *Animal* relation we achieve the highest F_1-measure (0.59), *Injury* and *Treatment* being comparable around 0.30 and *Result* yielding a far worse F_1-measure of just 0.13. In all but one case the systems output leads to higher precision than recall (with injury being the exception). In all cases but *Animal* we note a strong difference between both. Regarding runtime we needed about 37 minutes for automatic annotation of the 32 test papers.

4.3. Discussion

Our test results are quite satisfactory, especialls with respect to Area under Curve. However these results do not translate quite to an evaluation on new papers. We identify three main reasons for that: First we trained on only a small set of training papers. Second the classifiers might be trained very well but the relevant instances are not detected by NER. Finally it should be noted that the problem is inherently difficult: Even the numbers achieved here are an improvement over the baseline set in [Paassen et al., 2014] for all relations but *Result*. This relation, on the other hand, is also the most complex semantically. Another improvement over previous work is the fact that the number of annotations found by the system is comparable to the number of expert annotations, while the system previously annotated around 4 times more.

5. CONCLUSION AND OUTLOOK

We described the challenge of extracting relational descriptions about preclinical experiments on spinal cord injury from scientific literature. To tackle that challenge, we introduced a cascaded approach of named entity recognition, followed by relation extraction. Our results show that the problem of relation extraction alone can be approached effectively using multiple classifiers and simple heuristics. However, we found that the current configuration does not scale ideally to new and unknown papers.

Future work includes active learning approaches to select additional training examples that are valuable to the training, major improvements in runtime of ontology-based NER and an extension and revision of the training set by medical experts.

In the near future we will release our system as a fully-functional annotation pipeline [10]. This lays the groundwork for populating a full-blown semantic database on preclinical studies of SCI treatment approaches as described in [Brazda et al., 2013], hopefully leading to a fruitful transfer from preclinical to clinical knowledge in the future.

[10] http://openresearch.cit-ec.de/projects/scie

	Training					*Test*				
Relation	**Points**	**Prec.**	**Recall**	**F_1**	**AuC**	**Points**	**Prec.**	**Recall**	**F_1**	**AuC**
Animal	416	0.83	0.81	0.82	0.90	100	0.91	0.84	0.88	0.94
Injury	316	0.77	0.86	0.81	0.85	80	0.88	0.85	0.86	0.90
Treatment	201	0.83	0.87	0.85	0.90	51	0.79	0.92	0.85	0.90
Result	379	0.80	0.91	0.85	0.92	95	0.89	0.82	0.86	0.94

Table 3: The system's relation extraction performance evaluated on a corpus of 18 papers. The number of data points, precision, recall, F_1-measure and area under the ROC (AuC) is reported for training (left) and test setting (right).

6. REFERENCES

[Björne and Salakoski, 2011] Björne, J. and Salakoski, T. (2011). Generalizing biomedical event extraction. In *Proceedings of the BioNLP Shared Task 2011 Workshop*, BioNLP Shared Task '11, pages 183–191, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Brazda et al., 2013] Brazda, N., Kruse, M., Kruse, F., Kirchhoffer, T., Klinger, R., and Müller, H.-W. (2013). The CNR preclinical database for knowledge management in spinal cord injury research. *Abstracts of the Society of Neurosciences*, 148(22).

[Buitelaar et al., 2008] Buitelaar, P., Cimiano, P., Frank, A., Hartung, M., and Racioppa, S. (2008). Ontology-based information extraction and integration from heterogeneous data sources. *Int. J. Hum.-Comput. Stud.*, 66(11):759–788.

[Damerau, 1964] Damerau, F. J. (1964). A technique for computer detection and correction of spelling errors. *Commun. ACM*, 7(3):171–176.

[Doddington et al., 2004] Doddington, G., Mitchell, A., Przybocki, M., Ramshaw, L., Strassel, S., and Weischedel, R. (2004). The Automatic Content Extraction (ACE) program: tasks, data, and evaluation. In *Proceedings of LREC 2004*, pages 837–840.

[Doms and Schroeder, 2005] Doms, A. and Schroeder, M. (2005). GoPubMed: exploring PubMed with the Gene Ontology. *Nucleic Acids Res*, 33(Web Server issue):W783–W786.

[Fan et al., 2008] Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. (2008). Liblinear: A library for large linear classification. *J. Mach. Learn. Res.*, 9:1871–1874.

[Ferrucci and Lally, 2004] Ferrucci, D. and Lally, A. (2004). Building an example application with the unstructured information management architecture. *IBM Systems Journal*, 43(3):455–475.

[Filli and Schwab, 2012] Filli, L. and Schwab, M. E. (2012). The rocky road to translation in spinal cord repair. *Ann Neurol*, 72(4):491–501.

[Franceschini et al., 2013] Franceschini, A., Szklarczyk, D., Frankild, S., Kuhn, M., Simonovic, M., Roth, A., Lin, J., Minguez, P., Bork, P., von Mering, C., and Jensen, L. J. (2013). String v9.1: protein-protein interaction networks, with increased coverage and integration. *Nucleic Acids Res*, 41(Database issue):D808–D815.

[Hakenberg et al., 2011] Hakenberg, J., Gerner, M., Haeussler, M., Solt, I., Plake, C., Schroeder, M., Gonzalez, G., Nenadic, G., and Bergman, C. M. (2011). The gnat library for local and remote gene mention normalization. *Bioinformatics*, 27(19):2769–2771.

[Hanisch et al., 2005] Hanisch, D., Fundel, K., Mevissen, H.-T., Zimmer, R., and Fluck, J. (2005). Prominer: rule-based protein and gene entity recognition. *BMC Bioinformatics*, 6 Suppl 1:S14.

[Hofmann-Apitius et al., 2008] Hofmann-Apitius, M., Fluck, J., Furlong, L., Fornes, O., Kolarik, C., Hanser, S., Boeker, M., Schulz, S., Sanz, F., Klinger, R., Mevissen, T., Gattermayer, T., Oliva, B., and Friedrich, C. M. (2008). Knowledge environments representing molecular entities for the virtual physiological human. *Philos Trans A Math Phys Eng Sci*, 366(1878):3091–3110.

[Ji and Grishman, 2008] Ji, H. and Grishman, R. (2008). Refining event extraction through cross-document inference. In *Proceedings of ACL-08: HLT*, pages 254–262, Columbus, Ohio. Association for Computational Linguistics.

[Jimeno et al., 2008] Jimeno, A., Jimenez-Ruiz, E., Lee, V., Gaudan, S., Berlanga, R., and Rebholz-Schuhmann, D. (2008). Assessment of disease named entity recognition on a corpus of annotated sentences. *BMC Bioinformatics*, 9 Suppl 3:S3.

[Lafferty et al., 2001] Lafferty, J., McCallum, A., and Pereira, F. C. N. (2001). Conditional Random Fields:

Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of ICML 2001*, pages 282–289. Morgan Kaufmann.

[Lemmon et al., 2014] Lemmon, V. P., Ferguson, A. R., Popovich, P. G., Xu, X.-M., Snow, D. M., Igarashi, M., Beattie, C. E., and Bixby, J. L. (2014). Minimum Information About a Spinal Cord Injury Experiment (MIASCI) – a proposed reporting standard for spinal cord injury experiments. *Neurotrauma*. in press.

[Lipscomb, 2000] Lipscomb, C. E. (2000). Medical subject headings (mesh). *Bull Med Libr Assoc*, 88(3):265–266.

[Lok, 2010] Lok, C. (2010). Literature mining: Speed reading. *Nature*, 463(7280):416–418.

[Nadeau and Sekine, 2007] Nadeau, D. and Sekine, S. (2007). A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26.

[Nedellec et al., 2013] Nedellec, C., Bossy, R., Kim, J.-D., jae Kim, J., Ohta, T., Pyysalo, S., and Zweigenbaum, P., editors (2013). *Proceedings of the BioNLP Shared Task 2013 Workshop*. Association for Computational Linguistics, Sofia, Bulgaria.

[Paassen et al., 2014] Paassen, B., Stöckel, A., Dickfelder, R., Göpfert, J. P., Kirchhoffer, T., Brazda, N., Müller, H. W., Klinger, R., Matthias, G., and Cimiano, P. (2014). Ontology-based extraction of structured information from publications on preclinical experiments for spinal cord injury treatments. *Semantic Web and Information Extraction Workshop (SWAIE), 25th International Conference on Computational Linguistics (COLING)*.

[Pafilis et al., 2013] Pafilis, E., Frankild, S. P., Fanini, L., Faulwetter, S., Pavloudi, C., Vasileiadou, A., Arvanitidis, C., and Jensen, L. J. (2013). The species and organisms resources for fast and accurate identification of taxonomic names in text. *PLoS One*, 8(6):e65390.

[Prinz et al., 2011] Prinz, F., Schlange, T., and Asadullah, K. (2011). Believe it or not: how much can we rely on published data on potential drug targets? *Nat Rev Drug Discov*, 10(9):712.

[Saggion et al., 2007] Saggion, H., Funk, A., Maynard, D., and Bontcheva, K. (2007). Ontology-based information extraction for business intelligence. In et al., K. A., editor, *The Semantic Web*, volume 4825 of *Lecture Notes in Computer Science*, pages 843–856. Springer.

[Savova et al., 2010] Savova, G. K., Masanz, J. J., Ogren, P. V., Zheng, J., Sohn, S., Kipper-Schuler, K. C., and Chute, C. G. (2010). Mayo clinical text analysis and knowledge extraction system (ctakes): architecture, component evaluation and applications. *J Am Med Inform Assoc*, 17(5):507–513.

[Sayers et al., 2012] Sayers, E. W., Barrett, T., Benson, D. A., Bolton, E., Bryant, S. H., Canese, K., Chetvernin, V., Church, D. M., Dicuccio, M., Federhen, S., Feolo, M., Fingerman, I. M., Geer, L. Y., Helmberg, W., Kapustin, Y., Krasnov, S., Landsman, D., Lipman, D. J., Lu, Z., Madden, T. L., Madej, T., Maglott, D. R., Marchler-Bauer, A., Miller, V., Karsch-Mizrachi, I., Ostell, J., Panchenko, A., Phan, L., Pruitt, K. D., Schuler, G. D., Sequeira, E., Sherry, S. T., Shumway, M., Sirotkin, K., Slotta, D., Souvorov, A., Starchenko, G., Tatusova, T. A., Wagner, L., Wang, Y., Wilbur, W. J., Yaschenko, E., and Ye, J. (2012). Database resources of the national center for biotechnology information. *Nucleic Acids Res*, 40(Database issue):D13–D25.

[Schuemie et al., 2007] Schuemie, M., Jelier, R., and Kors, J. (2007). Peregrine: lightweight gene name normalization by dictionary lookup. In *Proceedings of the Biocreative 2 workshop 2007*, page 131–140, Madrid, Spain.

[Steward et al., 2012] Steward, O., Popovich, P. G., Dietrich, W. D., and Kleitman, N. (2012). Replication and reproducibility in spinal cord injury research. *Exp Neurol*, 233(2):597–605.

[Strassel et al., 2008] Strassel, S., Przybocki, M., Peterson, K., Song, Z., and Maeda, K. (2008). Linguistic resources and evaluation techniques for evaluation of cross-document automatic content extraction. In *Proceedings of the Language Resources and Evaluation Conference*, pages 2706–2709.

[Thomas et al., 2012] Thomas, P., Starlinger, J., Vowinkel, A., Arzt, S., and Leser, U. (2012). Geneview: a comprehensive semantic search engine for pubmed. *Nucleic Acids Res*, 40:W585–W591.

[Tsujii, 2009] Tsujii, J., editor (2009). *Proceedings of the BioNLP 2009 Workshop Companion Volume for Shared Task*. Association for Computational Linguistics, Boulder, Colorado.

[Tsujii et al., 2011] Tsujii, J., Kim, J.-D., and Pyysalo, S., editors (2011). *Proceedings of BioNLP Shared Task 2011 Workshop*. Association for Computational Linguistics, Portland, Oregon, USA.

[Wimalasuriya and Dou, 2010] Wimalasuriya, D. C. and Dou, D. (2010). Ontology-based information extraction: An introduction and a survey of current approaches. *Journal of Information Science*, 36(3):306–323.

INTELLIGENT SYSTEMS PROJECT: AUDEYE II

A. Kariryaa, M. Prasad, S. Sharma, O. Ast

Faculty of Technology, Bielefeld University
Bielefeld, Germany

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

ABSTRACT

AUDEYE II is an interactive, cognitive chess assistance system that supports its user by providing hints during a chess match. It is capable of tracking a chess game in real-time by recognizing chess board and position of figures using computer vision on the video input from a mobile eye tracking system. User can also interact with the system through eye gestures tracked by online eye gaze analysis. The system can give hints to user through sound output.

In this paper we describe the technique used for chess piece detection and improvements in chess board detection. Chess piece detection eliminates the requirement to play the game from starting as the system is no more dependent on the initial positions of the chess pieces (which was used to estimate the position of the game as the game progresses),so now the user can start using the system even from the middle of the game. Chess board detection technique helps to cope with only partially visible chess boards. Moreover we present a player classification module for analysing and tutoring the player.

1. INTRODUCTION

AUDEYE II system uses EYE-Tracking Glasses [1], that have a camera which tracks user's eye movements and provides a scene video with overlaid gaze cursor as well as a text file with timestamped gaze positions within the video coordinate system.

AUDEYE II system uses the video from the eye-tracking glasses and computer vision techniques to determine the chess board position and the state of the game. It then feeds this information to a chess engine which is used to track the position of the pieces and provide hints to the user if requested.

AUDEYE II is an extension to AUDEYE from previous year's ISY-Master students project [2]. AUDEYE can identify the chess board when it is completely visible in the video stream. It uses poker chips with chess figures drawn on it as chess pieces. The color of the poker chips (red or green) is used to determine the actual color (black or white) of the chess figures. Also the system has to track the chess pieces from the starting of the game since it is unable to detect the figures on the poker chips.

In this project we improved the original AUDEYE system by adding new features of the piece detection, board detection and player classification which are explained in detail in the next sections.

Section 2 describes the work which was done on the prototype so far. The following sections 3 to 5 explain the individual parts of the project (Piece detection, Board detection, Player classification) in detail, where each part is organized in concept, realization and results. Finally, section 6 discusses the results and future work, followed by conclusions in section 7.

2. PREVIOUS WORK

Our work is based on the AUDEYE prototype of last years' ISY-Master students [2]. The hardware setup is not changed and consists of an eye tracking glass connected to a computer, which provides a video from the players' view and eye gaze data. The software consists of several modules for piece and board detection, eye tracking, chess game extraction, voice output and eye gesture detection. The latter is used to communicate with the system. Four gestures were defined to enable/disable sonification, two gestures for training/testing and finally, one to request the next move. Here we hooked in to use the eye gazing data also for player classification and thus adapting to the amount of help given (i.e. move-by-move suggestion or recommend a strategy).

The video from the eye tracking glasses is used to follow the game. Several of the mentioned software modules extract the board and individual fields to determine the position of the chess pieces (for reasons of simplicity poker chips were used). From their color (red or green) the piece detection determines only which player owns the chip. This approach requires to monitor the whole game from the very beginning on. So, it was guaranteed that the chess extractor module could track the pieces move by move.

With our new approach we want to enable the system to not only detect the position of the pieces, but also which type of figure they are. This will improve the system's robustness to situations where moves are made without looking directly on the board (i.e. because you grin at you opponent while moving your last figure to set him checkmate) and also to be able to start the assistant at any point of the game.

Very crucial for this task is the board detection, which worked very well in AUDEYE when the chess board was visible in its full extend. The detected lines of the board are used to deskew the image so that the chess board extraction module can identify the individual fields which are used to detect which figures are placed on them. But the system stops working if the chess board is not fully visible in the video. Here our goal is to improve the detection, so that in can also handle only partially visible boards to keep track of the game.

3. PIECE DETECTION

Our vision is to identify 3D chess pieces through the video input from eye tracking glasses using computer vision. The task of identifying 3D chess pieces becomes quite difficult due to specular reflections and shadows which are of same color (black and white) as our pieces and the board (Fig. 1(a)). Also pieces in front of each other occlude other pieces which causes further problems.

So, we restricted our goals to identify 2D pictures of chess figures as shown in Fig. 2(a). In subsection one we describe the theoretical approach, followed by the realization in subsection 2. The evaluation of the new approach is done in subsection 3.

(a) Image of the original chess board shows the difficulty in identifying pieces due to shadows reflections and similarity between the color of board and pieces

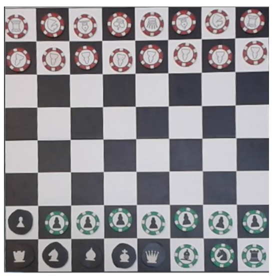

(b) Image of the test video setup including "old" (with green and red borders) and "new" (completely black and white) pieces

Figure 1: Chess boards used in this study

3.1. Concept

To identify pieces, we extracted the images of individual pieces from the whole chess board. We tested many algorithms, like matching SIFT features, good features to track,

(a) Image of the chess pieces

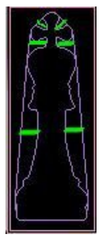

(b) Convex Hull drawn around it with defect pionts(depicted in green)

Figure 2: Figure detection using defectpoints on convex hull

template matching, etc. but non of these produced convincing results. Then we tried a different approach to identify pieces. The idea was to draw a convex hull around the image of the piece (Fig. 2(b)), then take the defect points highlighted in green in the Fig. 2(b) and compare these defect points with the sample images we had for the particular pieces. Since we had to do this matching for every piece present on the board (which can be up to 32) with all the 6 different types of pieces (king, queen, pawn, etc.) for every frame, the algorithm proved to be quite slow. So, we came up with another approach which proved to be much simpler and more efficient than the previous one.

3.2. Realization

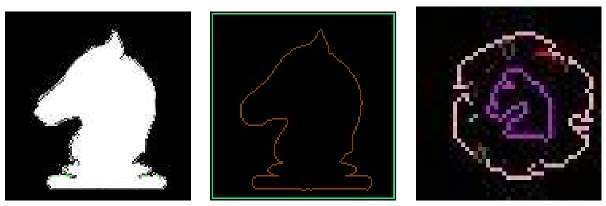
This time we used contour matching to identify pieces. We take sample images of pieces as shown in Fig. 3(a). Afterwards we apply a threshold and extract contours around it (Fig. 3(b)). This figure is then matched with the image extracted from realtime video input, which after extraction and several filters looks like Fig. 3(c). We compare the contours of two images using Opencv function cv2.matchShapes() to get the results. It is calculated based on the seven rotation-invariant Hu-moment values [3] of two contours A and B as follows (from [4]):

$$match(A,B) = \sum_{i=1...7} \left| \frac{1}{m_i^A} - \frac{1}{m_i^B} \right| \tag{1}$$

with

$$\begin{aligned} m_i^A &= sign(h_i^A) \cdot log(h_i^A) \\ m_i^B &= sign(h_i^B) \cdot log(h_i^B) \end{aligned} \tag{2}$$

and h_i^A, h_i^B being the Hu-moments of A and B. The lower the match value, the better the two contours match. Thus, the prototype whose contour has the lowest Hu-match-value is taken as figure type of the piece in the video. Fig. 4 shows



(a) Sample image of the chess piece to be matched (b) Contour of the image to be matched (c) Contour of the image from the video

Figure 3: Matching of sample figure of chess piece(knight) with video input based on Hu-movements

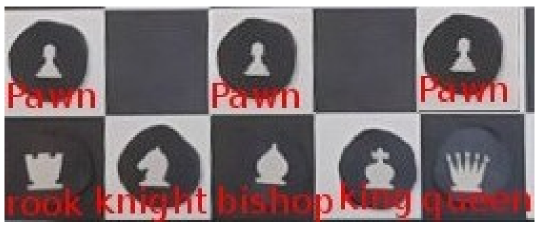


Figure 4: Results of the new detection algorithm on the live video stream from the eye tracking device

the debug output of the names of the pieces as identified by the system from an in-game video sequence.

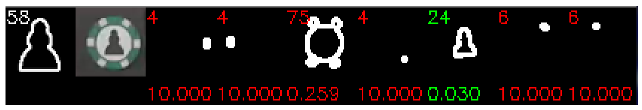
As in the future other pieces (also 3D pieces with advanced shapes) should be detected, we no longer get the player color by checking if the chip is green or red. Our approach is that the best matching contour is used as a mask on the binary image of the chip/piece. Depending on the amount of white or black pixels inside this mask, the player is selected.

Looking at the ”old” white pieces and their blurred appearance in the video during the first tests, we also created a test set of 6 ”new” white pieces. They have a better contrast between figure and background, so the contour can be extracted even from blurred images (compare Fig. 5 (c)-(f)).

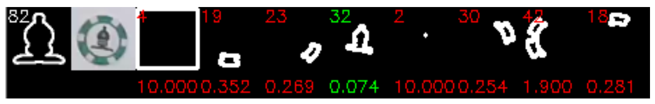
Using this method on the whole board, in every frame was likely to be way slower than the color-only approach, which was proved during the evaluation (see Tab. 1). Thus, the idea came up to use the already available information about the current state of the chess game, to boost the detection speed. As a result, we use the probability of a figure to be at a field as prior probability for the result of the contour matching.

3.3. Evaluation

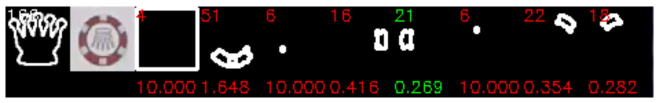
To evaluate our algorithm, we looked at the time the piece detection takes, compared to the color-only player detection approach and on the accuracy of the detection. The test was



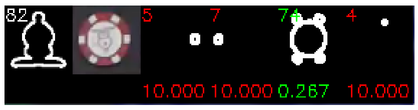
(a) Correctly detected contour of an ”old” black pawn



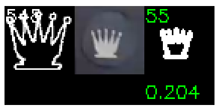
(b) Correctly detected contour of an ”old” black bishop



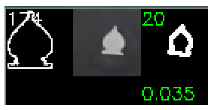
(c) Contours of ”old” white queen not extracted correctly



(d) Contours of ”old” white bishop not extracted correctly



(e) ”new” white queen detected correctly



(f) ”new” white bishop detected correctly

Figure 5: Each image shows the prototype piece on the left, then the input image to be matched and to the right the extracted contours. The numbers on the top are the amount of points per contour. If it is lower than 12 the contour is ignored (match value = 10). The value at the bottom is the result from the contour matching (see Eq. (1)). The green coloured value is the best match to the prototype.

setup on following system:

- Intel Xeon E3-1230v3 @3.30Ghz x 4
- 8 GB Ram
- Windows 7 Professional 64bit SP1
- HIS Radeon HD7770 GHz Edition, 1 GB Ram
- All tests were run single-threaded

To compare the time the detection of the whole board needs, we tested three modes:

1. Only ”old” color-based player detection
2. Figure detection on all fields without prior knowledge
3. Figure detection with prior knowledge only on occupied fields

The measurements were taken over several runs of the same 2:37 minute (430 frames) video of a chess board equipped with a complete set of ”old” chip-pieces for the

color only	all	known fields
4.01 ms	9.27 ms	2.10 ms

Table 1: Time to detect all pieces on the whole board (mean over 5 runs).

white player. The black pieces already had a good contrast between figure and background, so there would not be a big difference to the ”new” black pieces. Therefore we decided to replace some of them by 6 ”new” white pieces (Pawn, Rook, Knight, Bishop, King, Queen). Which leaves 10 ”old” black chip-pieces (7 Pawns, Bishop, Knight, Rook) (compare Fig. 1(b)). So we were able to compare the detection of all piece types in the same video sequence.

First we tested whether the rotation of the piece has any influence on the detection quality. As we used the ”matchShapes()” method from openCV [4] we expected no influence which was also confirmed by our data.

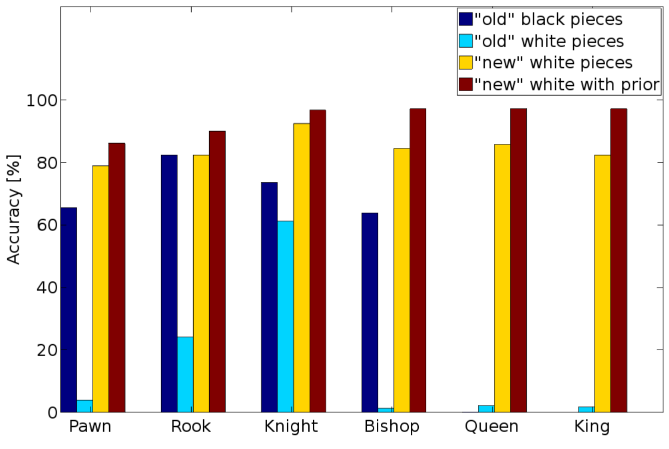
For the duration performance comparison we measured 5 runs of the whole above mentioned video (see Tab. 1) and calculated the mean time the piece detection needs, to detect all 32 pieces on a fully equipped board in one video frame.

Our goal was to improve the detection of the pieces on the board, so it is possible to start the assistant at any time during the game. From table 1 it becomes clear that a full board scan takes about 2 1/2 times longer than only detecting the piece color. Though this is a considerable extra time, the latter does not provide any information about the piece type and thus the current state of the game. Once the state of the board is known, the new piece detection is even twice as fast as the color-based detection because we use the knowledge of the current game state. So, we get a considerably higher amount of information in less time.

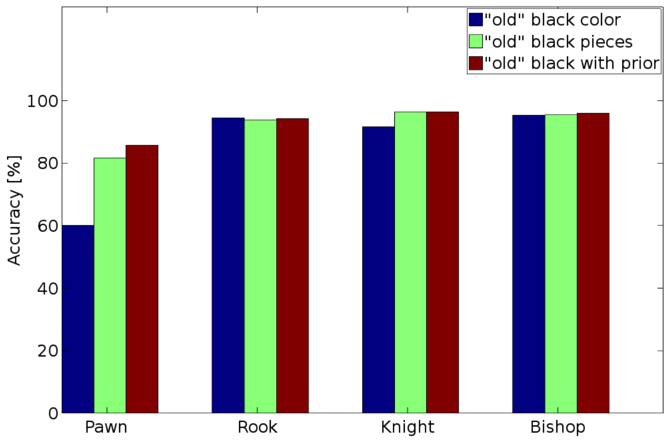
But how good is this information? The detection accuracy tests were done on two different piece prototype sets. One for ”old”, one for ”new” chips. Each was run five-times on the test video and the average correct detections over all frames was calculated (see Fig. 6). It clearly shows that it depends on the type of piece. The ”old” black pieces already get detected with 50 to 80 percent. The ”old” white pieces, in contrast, hardly are detected at all.

From the detailed analysis of the compared contours (Fig. 5), it becomes clear that the white figures' contours are not even extracted from the input image, due to its bad quality. This results from the limited focus of the camera and the processing to get an undistorted image. To overcome this, we designed new pieces with a higher contrast. With the new pieces the detection results increased dramatically to 80 to 90 percent accuracy (see Fig. 6(a)).

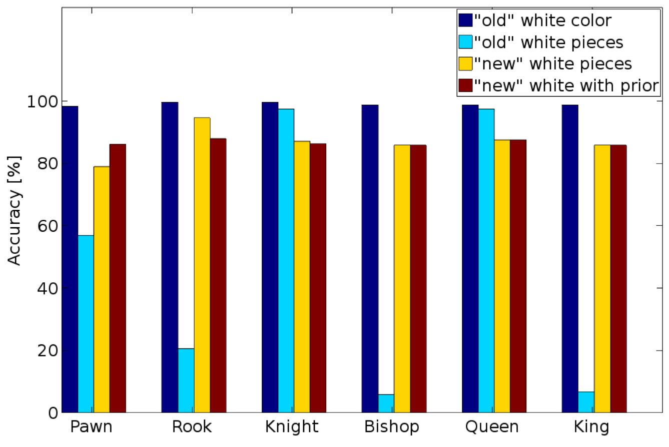
A similar result can be seen on the player detection accuracy (see Fig. 6(b) & (c)). The reason for some ”old” white pieces to be detected correctly, is because they are located on a black field and the resulting contours enclose a



(a) figure detection



(b) detecting pieces of black player



(c) detecting pieces of white player

Figure 6: Accuracy of detection. The value for pawn is the mean of all pawns of a type. There are no black king and queen on the board, due to the test setup with new white pieces (see Fig. 1(b)). The first column of the player detection is always the ”old” color detection.

mainly white area. On white fields the red ring of the chip is detected as mostly black (binary image) and thus results in the wrong player. With the new figures this is also no longer a problem, though detecting ”white” pieces is about 10% less accurate than with the color-only approach. The player detection on pawns works better than on the other figures, because their image only consists of a shape. This can be matched better than the structured images of i.e. the king.

So, with the new set of pieces the system can detect figure types more reliable and by using the prior knowledge of the board state the detection got twice as fast. Thus, the improved chess piece detection enables the other tasks like player classification and the chess-engine to work more efficiently.

4. BOARD DETECTION

4.1. Concept

In the previous Audeye system, the chessboard detection was using Hough transforms to detect lines in OpenCV [5]. The board will be represented by a perspective-distorted grid of 9x9 lines which can then be rectified. This method is not robust enough for the visualization of chessboard.

Chessboard detection makes use of the open source computer vision library in OpenCV. Although OpenCV provides a build-in function to detect chessboard 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. We initially used the standard OpenCV algorithm, Template matching; this method includes an automatic mechanism for counting the number of squares in the grid and predicting the grid corners. But this method is less robust and unable to detect all corners of the chessboard.

Localization of chess-board vertices is a common task in computer vision, relatively little work focuses on designing a specific feature detector that is fast, accurate and robust. Our aim was to achieve robust chess-board detection. The method proposed is robust against noise, poor lighting and poor contrast, requires no prior knowledge of the extent of the chess-board pattern, computationally very efficient, and provides a strength measure of detected features. This concept also includes the optimized chessboard detection even though if it is partially visible.

4.2. Realization

By implementing the approach proposed by W.Sun [6], we expected to detect the inner corner points of chessboard, which is robust enough to detect the partial chessboard. Each checkerboard grid is a solid intensity region where the internal corner is surrounded by four alternating bright and dark regions. By detecting and locating region corners(Fig 7) of checkerboard pattern surfaces in two scenes, the mapping between those corresponding corners on the deformed surfaces of the different scenes can be calculated. The mapping between those two deformed surfaces actually consists of many homography between the approximate planar quadrant grids of the checkerboard patterns in the different scenes.

This approach propose a robust automatic method that can recognize checkerboard pattern under complex illumination and deformation. The main idea is to treat a certain neighbour points within a rectangle or a circular window and transform the 2D points distribution into 1D points to detect regions. Corners are correlated and clustered by the region boundary data to recognize the checkerboard corners. The 2D point distribution is transformed within a window to 1D, then regions in a layer will be line segments after binarization(Fig 9). This will simplify the detection by recognizing the four alternating regions features in this method.

Ring-morphology(Fig 8) treats a 1D image as a ring that the head and the end are joined. Due to noise in each frame which cause a fake region as show in Fig. 10. By performing ring morphology opening and closing on the binary layers after thresholding, the fake regions can be reduced. Thresholding in 1D layers locally will get robustness against various illumination. The fake region reduction can ensure the correctness of the corner detection. The acceptance threshold of the layers and other parameters with the values defined in this method according to most general cases will reduce the effects caused by noise or deformation. Noise reduction in checkerboard match and the redundant corner reduction by clustering can ensure the corners without redundancy and noise. Then corners connected by region boundaries will produce the right checkerboard corners set.

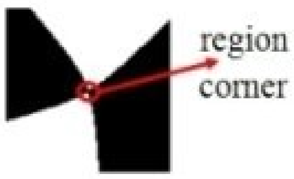


Figure 7: Region Corner

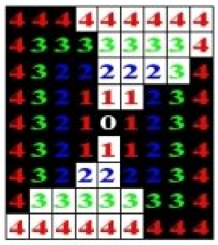


Figure 8: Ring Morphology

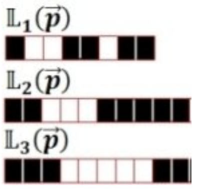


Figure 9: 1D form of layers

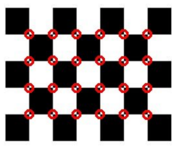


Figure 12: Checkerboard Detection

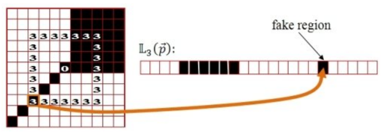


Figure 10: A fake region caused by a line

The last point and the first point of a layer in 1D form are actually continuous in 2D form, a region may break here because of the 2D to 1D transformation. An example is shown in Fig. 11. These broke regions may be treated as fake regions when noise reducing because of their short widths. To avoid this problem, the binary morphology operations by elongating the layer to repeat it one more time behind its last point to be cycle-compatible, these modified operations are called the ring-morphology. The result of this approach will detect the corner points in a checkerboard as shown in Fig. 12.

5. PLAYER CLASSIFICATION

5.1. Concept

To develop a better chess assistant system, the need to understand the player's chess expertise arose. Expertise in chess goes along with the recognition of chunks and templates in the given chess constellation [9]. In order to provide customized help and recommendations the system has to analyze and assess the player. The chess engines so far do not provide individualized support and guidance, our approach is to develop a tutoring system that identifies the weakness of the individual player and thus provide focused guidance. In this regard of developing a player classification mechanism two approaches were considered. In our first technique we perform a structural dimensional analysis of mental representation (SDA-M) [7] on chess constellations. Mental representations are characterized by well-integrated networks of basic action concepts or BACs. The concept of SDA-M was initially used in sports to measure the movement representation in long term memory but it has also used for other purposes like representation of know/unknown objects [8]. In sports involving movement, each BAC corresponds to functionally meaningful body postures and movement elements but in chess they would represent representation/structures of chess constellations. In this pregame assessment of the mental representation structure using the SDA-M technique, the test player is presented with a classification set, consisting of chess constellations, for clustering the BACs using the split program. In the split program the test player labels chess constellations as similar or dissimilar to the anchor constellation. This information i.e. distance between the various BACs is stored in a distance matrix. Then a hierarchical cluster analysis is used to transform the set of BACs into a hierarchical structure. These hierarchical clusters created by the test player are compared with the hierarchical structure of the BACs from an expert (Fig. 13(a)), depending on the similarity value of two structures the player is given a rank or classified broadly.

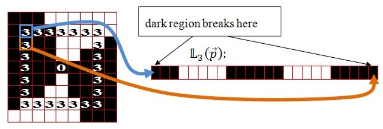


Figure 11: Broken region of the region corner

The second approach is the online eye gaze assessment, using the eye gaze movements recorded with the eye tracker. The data from the eye tracker could be analyzed in several possible ways. One option is to present the player with a chess problem to solve and then compare the eye gaze movement of the player in the situation with that of the expert. The other possibility is to analyze the eye gaze movement of the player in real time with the help of the chess engine. For this, the most probable move of the player could be guessed by analyzing the eye gaze movement and then the move could be ranked using the chess engine. Saccades between the most fixated field could also be analyzed to find the most probable move of the player. The rank of move, then, could be used to further assist the player.

The two approaches, pregame assessment of mental representation structure and online eye gaze assessment, are then merged and compared either with each other or with a standard player classification technique for chess. This allowed us to investigate the effectiveness and authenticity of the these two approaches. Furthermore, the results of the two approaches can also be used to analyze the weaknesses of chess players and provide adequate training.

5.2. Realization

For the pregame assessment of the mental representation structure, two different sets of stimuli were developed for the split program. The two sets had different criteria, to analyze the critical thinking ability for finding the most probable move and to analyze the critical thinking ability for finding the end result of the game The criteria for one set was to find the balance in the game and determine whether white or black was in the better position in the given chess constellation. Each chess constellation required different level of assessment, in some constellation the situation was clear and in others, some specific and definite development in game puts either black or white in a strong position. The test player was expected to find such situation and then decide upon the side which was stronger. Fig. 13(a) and 13(b) shows example stimuli of chess constellations used in the experiment. In Fig. 13(a) black has advantage over white or can produce a definite advantage and in Fig. 13(b) white has an advantage or can produce a definite advantage over black, hence these two chess constellations are highly dissimilar.

In the other set, the player had to classify the chess constellation on a different parameter. Here white had a clear advantage over black and the player was expected to determine the number of moves in which a definite checkmate could be achieved.

On comparing the result of the test player with the expert player, the test player's understanding of the game was determined and player was classified in either Novice, Medium or Expert category. This classification procedure was tested with 8 players, 3 of them were experts from the Bielefeld Chess club and Bielefeld University, where as rest were non expert chess players from the university.

In Online Eye Gaze assessment the test player was asked to find the best possible move in the presented chess constellation. The eye gaze movements were recorded using the eye tracker (here we used Eyelink II [10] for improving the performance of the system as Eyelink II has a higher frequency(250 Hz) than the ETG glasses(60 Hz) and hence we had more precise information and better results). This data was used to calculate the parameter like average fixation duration per field, Number of fixations per field etc. From these parameters the key figures and the key squares were calculated. These were compared with key squares and figures calculated from the expert's eye gaze movements to determine the player's understanding of the problem. Then as part of the player tutoring system, player is advised to focus on the key figures and key squares and look for the best move using this information. This classification procedure was tested with 4 players, 2 of them were experts from the Bielefeld University, whereas rest were non expert chess players from the university. Some stimuli had similar constellations with different level of distractions and expert avoided the distraction to focus on the problem but novice player could be distracted and hence fixated upon less important pieces and squares.

5.3. Evaluation

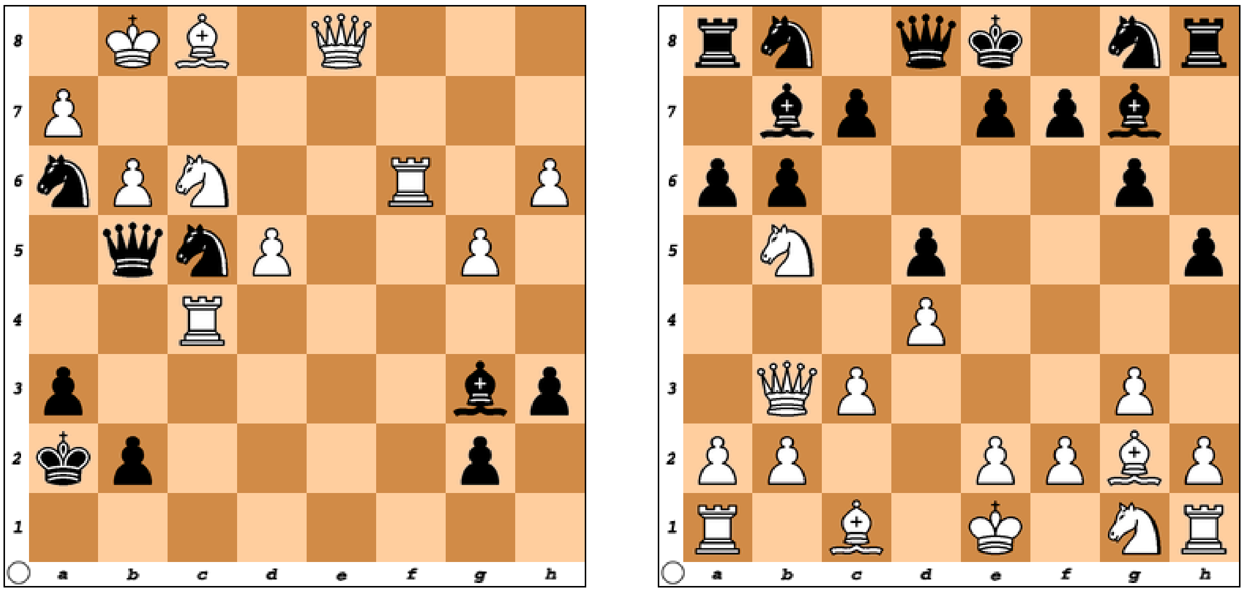
In Fig. 14 , you can see the result of pregame assessment of the mental representation structure of a test player and an expert. In the Fig. 14 , we can see that the fine classification by the expert and the coarse classification by the test player. A high similarity value (close to 1) represents a higher expertise level. The result of the SDA-M Procedure are dendograms that are more hierarchically in case of experts and less in case of novices. On the y-axis we see the Eucledean distance between the stimuli (the more often two stimuli are considered as similar by the test player, the lower are their Euclidean distance).

The result from the online eyegaze analysis is presented in the Fig. 15(a) and Fig. 15(b). The Total fixation duration per field and the Number of fixation per field are presented in the Fig. 15(a) and 15(b) . It is clearly visible from the eye gaze movement of the expert that row 8 and column D have high significance in this particular situation and the rook at D2 is a key figure. The eye gaze movement of the test player closes resembles that of the expert in this case. The high number of fixations on the square D3 could be because of the experimental error and drifts in fixations. It can also be observed that the expert had a more focused vision of the game than the player.

The result from the two approaches have not yet been combined together or compared with the result from a standard chess classification technique.

6. DISCUSSION AND FUTURE DIRECTION

We improved the chess piece and board detection and they have produced promising results. Also we are the first who worked on individualized feedback using an eyetracker. But even under simplifications, the computer vision challenges we are facing in this project are inherently difficult to perform. It turned out, that even with knowledge about the objects, we still need constraints to develop a feasible system in terms of usability. The new feature of player classification with the pregame assessment of mental representation structure and online eye gaze assessment has produced good results. However, our approach first needs further evaluation and comparison with the standard player ranking methods. In the future, online eyegaze assessment could be improved with involving the assessment from the chess engine.



(a) White to move, black has advantage (b) White to move, white has advantage

Figure 13: Stimuli from the classification set of finding the balance in the game

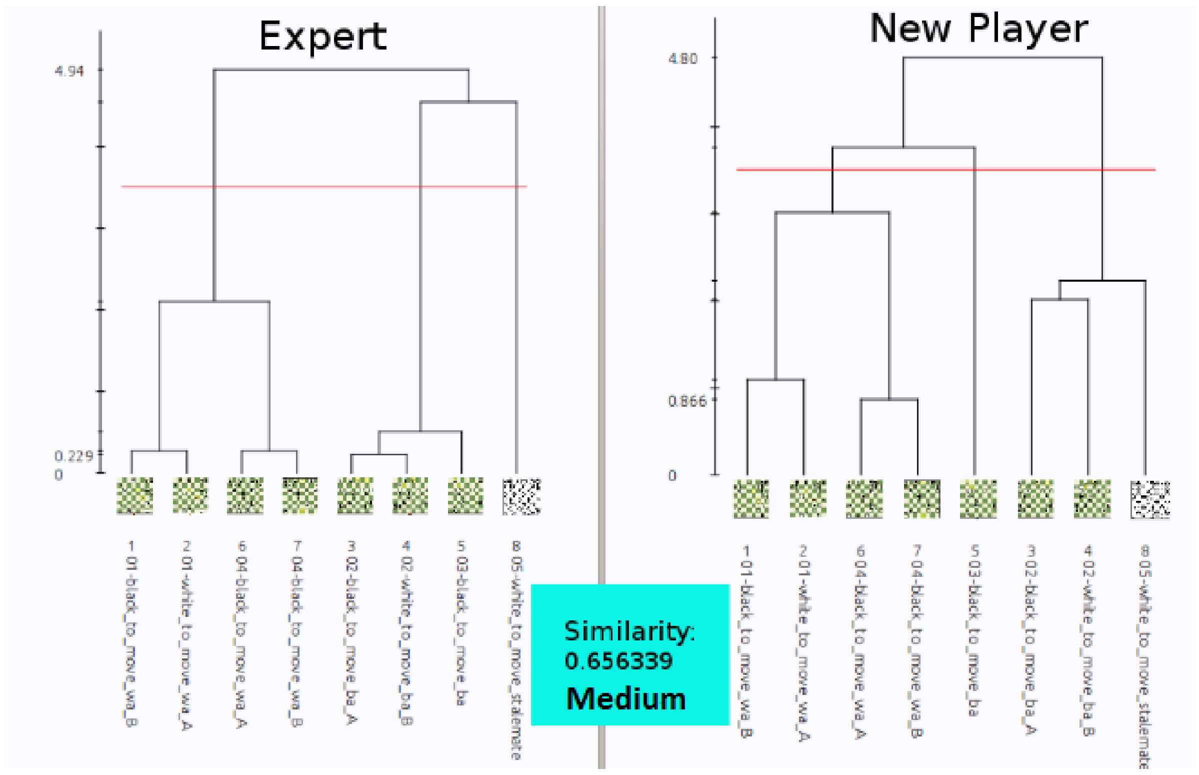
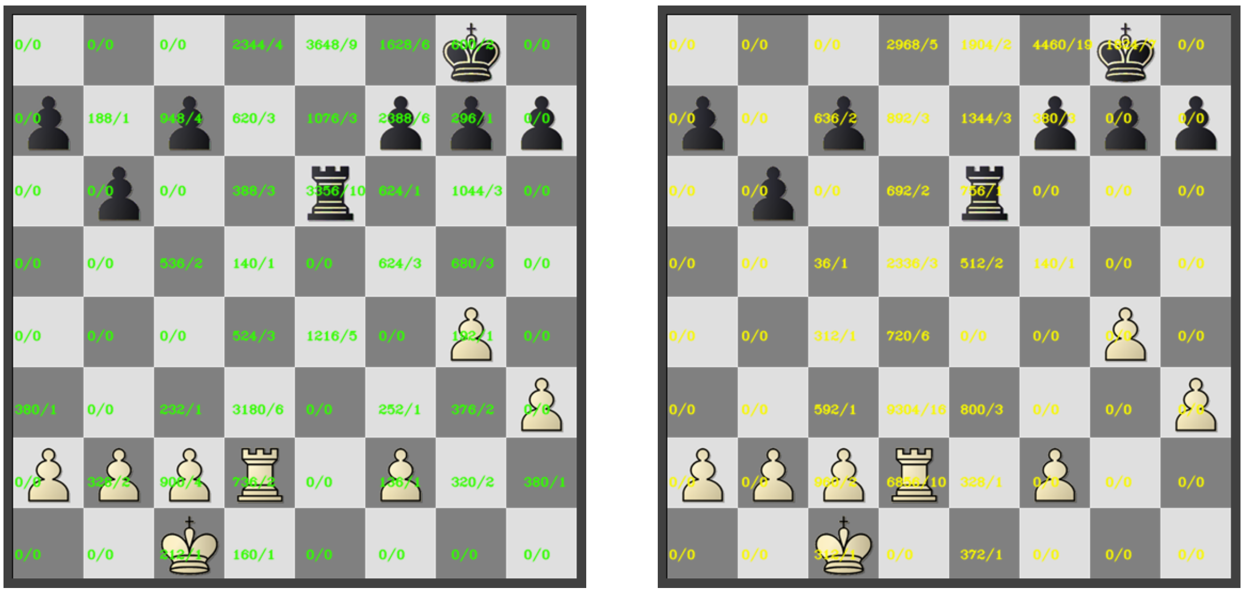


Figure 14: Pregame assessment of the mental representation structure



(a) Test Player (b) Expert Player

Figure 15: Eye gaze results

The future work should concentrate on increasing the comfort of the player further and to create an invisible, easy-to-use assistance tool. Further more a player tutoring system could also be added along with the assessment system.

7. CONCLUSION

Our previous system was able to assist chess game by providing hints for next moves to the users but was unable to detect chess pieces and chess board detection was also not very robust. We were able to successfully detect the 2D figures chess pieces so now the system can detect the positions of chess pieces even when the glasses where not used from the starting of the game.We were also able to classify the player's expertise level based on the the eye movements and mental representation structures and long-term memory.

8. ACKNOWLEDGEMENT

We thank our supervisors Dr. T. Pfeiffer and Dr. K. Essig for their great support during the project.

We also thank Thomas Küchelmann for his support in creating the classification set and Florian Paneff for helping in evaluating the classification set.

9. REFERENCES

[1] SMI Eye Tracking Glasses by SMI Berlin `www.smivision.com`

[2] V. Losing, L. Rottkamp and M. Zeunert, "AUDEYE," in Proceedings of the ISY Workshop, Issue #1, pp. 10–16, Bielefeld, Germany, July 2013.

[3] J. Flusser: On the Independence of Rotation Moment Invariants, *Pattern Recognition*, vol. 33, pp. 1405-1410, 2000

[4] OpenCV matchShapes Documentation `http://docs.opencv.org/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html?highlight=cvmatchshapes#matchshapes`

[5] OpenCV Hough Line Transform `http://docs.opencv.org/doc/tutorials/imgproc/imgtrans/hough_lines/hough_lines.html`

[6] Weibin Sun, Xubo Yang, Shuangjiu Xiao and Wencong Hu, Robust Recognition of Checkerboard Pattern for Deformable Surface Matching in Multiple View

[7] T. Schack, Measuring Mental Representations, pp. 203-214, 2012

[8] Essig, Strogan, Ritter and Schack, Influence of Movement Expertise on Visual Perception of Objects, Events and Motor Action- A Modeling Approach, pp. 2-15, 2012

[9] Chess Player Assessment `http://en.wikipedia.org/wiki/Wikipedia_talk:WikiProject_Chess/FAQ/Assessment`

[10] Eyelink II Technical Specifications `http://www.sr-research.com/pdf/elII_table.pdf`

INTELLIGENT SYSTEMS PROJECT: SMART RESCUE ROBOTS

Kai Harmening, Martin Holland, Andreas Langfeld

Faculty of Technology, Bielefeld University
Bielefeld, Germany

Supervisors: Rene Griessl, Stefan Herbrechtsmeier, Timo Korthals, Prof. Dr.-Ing. Ulrich Rückert

ABSTRACT

Supporting human rescue teams with robots is a current research topic [4], [9]. We focused on the task to identify human victims and dangerous areas for rescue teams. Our work is using a multi robotic approach in which cheap small robots shall assist the rescue squads. We built a heterogeneous distributed multi sensor system that classifies objects, in which each robot has only one sensor and is called expert for his sensor.

BeBots [8], developed at Heinz-Nixdorf-Institute at Paderborn are of use as basis for this task. As *Middleware* to realize communication between the BeBots *RSB* [16], Robotics Service Bus developed at CorLab Uni-Bielefeld, was introduced. Each expert robot uses its sensor to classify a human. In regards to a multi sensor system the results of each expert were combined in a sensor fusion, which computes the final result. Our evaluation shows that each expert works seperatly and the fusion is able to classify on the basis of the experts output whether there is a human or fire.

1. INTRODUCTION

In a disaster area one main task of rescue teams is to recover and safe victims. This is a dangerous task due to the fact that rescue teams often have to deal with hazardous and unknown situations. In this application area, robots can be used to support them. The advantages of using robots in rescue scenarios are threefold [9]. First, the robots could explore new areas and search for further victims while the rescuers safe current known victims. This kind of task sharing increases the posibility of finding and saving all victims. Second, robots are superior to humans in hazardous and long term situations, due to the fact that only the life of battery is limiting. Finally, a robot is less valuable than a human and can be replaced more easily. Instead of a human rescuer, only robots have to risk their “life”.

One challenge they could adopt is determining the position of victims. For this task the robot must be able to classify a human and drive to the place.

The multi robotic approach aims at the replacement of complex single solutions, like the GETBot [13], developed at the University of Paderborn, to reach a higher overall system robustness. Another advantage is that small robots are not as expensive as a big robot.

In this paper, we present a sensor fusion with focus on the sub-task of classifying humans in rescue scenarios by a distributed, heterogeneous sensor system. This means that we have different sensors and each of them is running on one robot. The classifiers are able to classify a feature of a human, for example human voice via microphone.

The sensor fusion is able to classify a human with the classification results of each sensor expert. So the classification of different human features is combined in the sensor fusion.

The design of our system is described in Section 2. Section 3 gives an overview of the construction of the hardware. The software components are explained in Section 4. Section 5 contains the evaluation of each classifier and shows the robustness of them. The discussion follows in Section 6.

2. SYSTEM DESIGN

The aim of this system was to classify objects, especially humans. Our approach was to use BeBots, which come along with several different sensors. Using all of these sensors simultaneously in approximately real time seemed impossible because of the limit computing power. We decided to split up each sensor and let each BeBot just use the input values of one special sensor. This BeBot became an expert for the sensor it used. So the BeBot which used for example the camera just used the camera for getting input data. To classify a human, it could do a face detection with the camera image and then sent the result of this classification to a further BeBot. This further BeBot was not an expert for any sensor, but it was an expert for the sensor fusion. It received and collected the classification results of all experts and fused them to a final classification (Figure 1).

The goal of this system was not to introduce any movement of the robots, even if this would bring new scopes. We focussed on the communication and classification. A possible application scenario is shown in Figure 2.

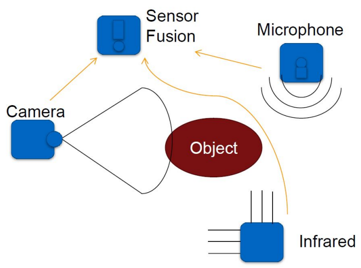
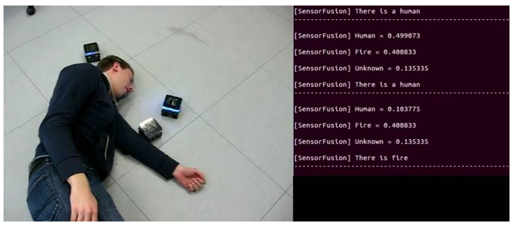


Figure 1: Sketch of the scenario. The yellow arrows represent the communication messages sent from the robots which are sensor experts to the robot which runs the sensor fusion



```
[SensorFusion] There is a human
--------------------------------------------------------
[SensorFusion] Human = 0.499073
[SensorFusion] Fire = 0.400833
[SensorFusion] Unknown = 0.135335
[SensorFusion] There is a human
--------------------------------------------------------
[SensorFusion] Human = 0.103775
[SensorFusion] Fire = 0.400833
[SensorFusion] Unknown = 0.135335
[SensorFusion] There is fire
--------------------------------------------------------
```

Figure 2: Application scenario. On the left side there is a victim which gets classified by the BeBots around it. On the right side there is the output of the sensor fusion.

This distributed classification had several benefits: we were not limited to just use the BeBots. We could combine different robots in this network, for example the AMiRo [7]. Also we were not limited to the sensors: a new sensor could be integrated very easily to the system by just sending its classification results to the sensor fusion, which could handle new input data adaptively. We were also independent to the number of robots in the system and to the number of robots using the same sensor, as mentioned in section 3.2.

3. HARDWARE & SETUP

In the following section we present the robot, called BeBot we used to build up a distributed multi sensor system. Also the communication interface we created, using the middleware Robotics Service Bus, short RSB is explained.

3.1. BeBot

As mentioned above we used BeBots [8] developed at Heinz-Nixdorf-Institute Paderborn, for our distributed multi sensor system. These robots, see Figure 3 are equipped with a low resolution SVGA-camera, a microphone and twelve infrared sensors around the chassis.

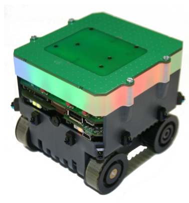


Figure 3: A picture of the front view of a BeBot

The BeBots operating system is Ubuntu, a pokey Linux operating system. The main processor works with 600 MHz and has got 256 MB memory. The robots also have a FPGA-Board to speed up computation of image processing tasks, due to the lack of CPU power. The BeBots provide Bluetooth, WLAN and USB connection for programming and communication. Because of the energy efficient design they can run up to 4 hours with one 3.7 V / 3900 mAh lithium-ion accumulator in our scenario.

3.2. RSB Communication Interface

We used RSB (Robotics Service Bus) [16], developed at CorLab Uni Bielefeld as middleware. To enforce a sensor behavior act model on the BeBot, which gave us the possibility to a simple integration of other robots as the AMiRo [7]. We needed two different kinds of RSB communication: an intra robot bus communication to send messages from one program to another on the same BeBot, and an inter robot communication, to send messages via WLAN to another BeBot (Figure 4).

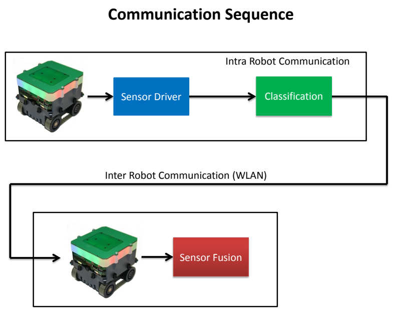


Figure 4: Overview of the communication sequence.

For the implementation of a modular sensor behavior hierarchy, we separated the sensor driver in one program and the classification of these sensor input data in another program. If the sensor driver had new input data, these data were sent via RSB on the intra robot communication to the affiliated classification. After the classification was finished just a float representing the percentage value of the classification was sent via the inter robot communication to a further BeBot. This robot ran two applications. First it ran a socket server which spread the network between the robots. As a second it ran the fusion program in which all different classification values were collected and fused them for a final classification of the output of all sensor classifications (Figure 5).

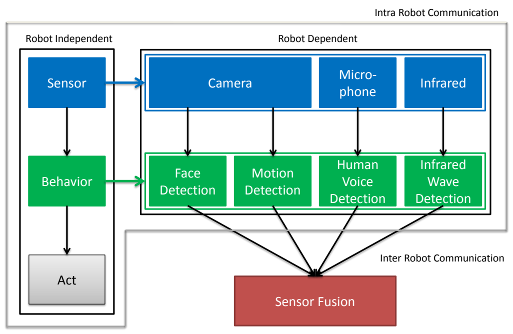


Figure 5: Overview of the whole communication structure.

The sensor fusion could distinguish all the different classification values it received as input by their specific sub scope. For this implementation we developed a hierarchical sub scope system, in which we combined the information about the BeBot who did the classification and the class to which the sent value belonged to in one sub scope. For example, if the infrared classifier, ran on BeBot number 12, calculated a fire-value of .87, this BeBot would sent the value .87 to the sub scope /IR/12/fire. So the fusion could distinguish to which class the sent value belonged to. This structure provided also the possibility to have more than one expert of the same kind in the network.

4. FEATURE DETECTION AND CLASSIFICATION

In the following chapter we explain how each sensor expert classifies features out of its special sensor input. We also describe the sensor fusion, which combines the features from each expert to a final classification whether there is a human or fire.

4.1. Microphone

The expert for the microphone is able to detect human voice in the captured sound. Due to the fact that we only wanted to classify the existence of speech we had to find characteristics of human voice. The characteristic we used was frequencies which are typical for human voice. To get the frequencies of our captured sound we used Fourier transformation [2]. Because of the limited computation power of the BeBot we implemented the “Danielson-Lanczos” algorithm [12]. The frequencies between 250 Hz and 2000 Hz contains the most significant frequencies of human voice. As our feature, which is the input of the classifier, we calculated the sum of the frequencies from 250 Hz to 2000 Hz divided by the overall frequencies.

Then we had to train our classifier with training data. The training data contained sound samples with and without human voice. After the training phase we got two functions:

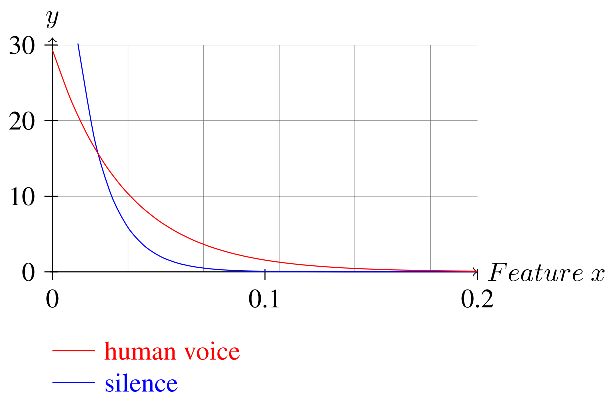


Figure 6: Two functions represents the classes “human voice” and “silence”

one for the classification result of human voice and one for the classification result of silence (Figure 6). The maximum value of both functions for feature x represents the class the output belongs to. Feature values lower than the point of interception represents the class “silence” and those which are higher represents “human voice”.

The sound we were classifying in our classifier is captured with the following recording settings:

- Frequency: 8000 Hz
- Channels: 1
- Format: “raw”

For capturing audio via microphone we used the program “arecord” of the alsa-utils [14]. We captured the lowest possible duration of one second into a sound file which has to be load by the classifier. From that sound file we used a small snippet of 256 Bytes.

4.2. Camera

For the assessment of camera images we implemented two different kinds of classification. On one hand there was a face detection, on the other hand there was a motion detection. We split these classifications in two programs. At this point, having a driver for the camera image was indispensable, otherwise both programs would try to capture the camera for their own if they are executed simultaneously. A camera driver publishing the images in the intra robot network prevented this, so both programs could simultaneously receive the current camera image and could do a classification. We used OpenCV [3] for the image processing.

4.2.1. Face Detection

We implemented two very different algorithms for the classification of a human face in the camera image, the LBPH (Local Binary Pattern Histogram) [1] and Eigenfaces [15].

LBPH extracts Local Binary Pattern histograms from small areas of the face and concatenates these histograms “into a single, spatially enhanced feature histogram”[1], which represents the face image. The classification comes about a nearest neighbor measure.

A different approach is Eigenfaces. In this connection an average face represented as a high-dimensional space is calculated. The eigenvectors of this space are called “Eigenfaces”, which represent “significant features”[15]. An image to be examined is mapped in this space. For the classification a weighted sum of this Eigenfaces is calculated, the resulting weights decide if there is a face in the image or not.

Both algorithms based on the same training data. We used a selection of the cropped face database of the University of Sheffield [11] containing 59 images with 20 different people of different age and gender. We just use their frontal face from different angles.

4.2.2. Motion Detection

Motion detection was just possible if the BeBot itself did not move at all. It was based on a mixture of Gaussian. For this it took a history of the last images and detected the differences in these images. Basically this was used to detect a foreground of an image but this foreground could be interpreted as motion in the image. For the classification, we counted the number of pixels belonging to the foreground and normalized them.

Due to the fact that this classifier could detect a foreground in an image in whom there was no movement, we needed an offset to prevent this. This offset represented the fixed foreground in the current basic frame and was calculated with no motion in front of the camera. After the calculation of the offset, the BeBot could detect motion only in the current scene. If the BeBot was moved to another viewpoint the old offset became invalid and it needed a new calibration to calculate the offset for the new scene. This process allowed the possibility to change the scenario with just a new calibration.

4.3. Infrared

The basic idea of the infrared classifier was to differ between three classes of infrared (ir) values. We had the idea, that there should be different distributions of the infrared values for each class, so that we could use estimators in order to get a possibility for each class, see Figure 7. The first class is “noise”, where nothing is close to the BeBot and the ir-values are very low. The second class is “human”, where the ir-values should be greater and the third class is “fire”, where the ir-values should be extreme high. In order to get possibilities between zero and one we normalized the output of the estimators used for the infrared classification.

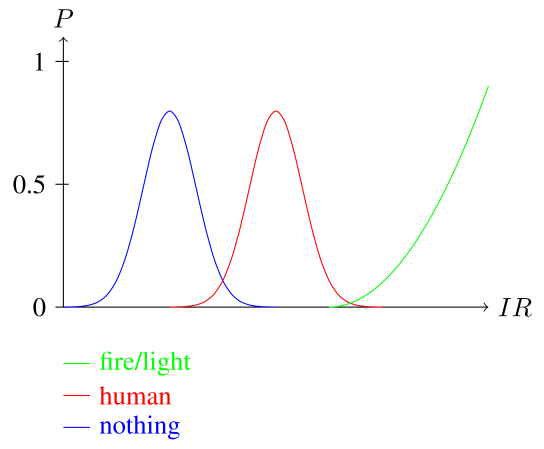


Figure 7: Diagram of the infrared value distributions. On the x-axis the infrared values from the sensor and on the y-axis the probability for the classes

The infrared sensors of the BeBot have a receiving and a sending component, which is a typical setup for getting information about distances to obstacles. For the task of measuring distances an offset is computed by the micro controller. This is done to get reliable data, that are cleaned of measuring errors, which can occur due to exterior influences like sunbeams or other light sources that the sensors can measure. This setup is normally used to implement collision behavior on the robots.

With this standard setup we were able to classify “noise” with a Gaussian estimator, because we got a normal distribution of the infrared values. The classifier was trained with example data recorded on a BeBot standing on an empty desk.

To classify “fire” or a “human” we needed a different setup of the ir-sensors. We wanted to get precise information about the intensity of the incoming infrared waves without any precomputation or offset that cleans the data. In order to differ between a human or a light source we had to reconfigure the BeBots micro controller that computes the infrared values from the raw data of the single sensors. With this data from the ir-sensors we thought it might be possible to get different distributions to classify whether there is a light source that sends ir-waves or there is a human. We got an exponential distribution of the infrared values for the light source and could classify on the basis of an exponential estimator for the class “fire”. The estimator needs training data of typical infrared values, very high values, when there is a light source.

To achieve the goal of classifying a human, the idea was to use the same Gaussian estimator that we used for classifying “noise” with the infrared expert. With the difference that we had to train it with a data distribution that appears when there is a human close to the sensor. We thought the distribution of the values would be normal distributed for this scenario. But the classification of ir-waves of humans failed, because the ir-sensors on the BeBot work on a limited spectrum wavelengths and the wavelengths of infrared light that comes from a human is just outside the range of the BeBots sensors. So we got no peak or distribution at all for humans.

4.4. Sensor Fusion

The sensor fusion is the node that has a connection to all other experts of our distributed multi sensor system. It is connected via our communication interface, described in chapter 3.2 to the experts that use a sensor to classify. The sensor fusion collects the results the experts computed for the three different classes “fire”, “human” and “noise”. These results are possibilities with floating point values between zero and one. Each sensor expert sends all features he is able to compute to the sensor fusion. That means the fusion BeBot gets results for “human” from the camera and microphone experts, for “fire” from the camera and infrared experts and for “noise” from camera and microphone experts. We use a Naive Bayesian approach [5] to fuse these sensor results. We basically use a maximum a priori classifier to compute a final result what the robots have detected, see (1). $P(S = s|C = c)$ describes the probabilistic information in sensor S about class C. The Bayes theorem provides the possibility for $P(C = c|S = s)$, from which the class with the maximal value is taken as winner.

$$\begin{aligned} C_{MAP} &= \arg\max P(C = c|S = s) \\ &\propto P(S = s|C = c)P(C = c) \end{aligned} \tag{1}$$

Because the experts work with different classification frequencies, due to computational power of the different tasks, it was necessary to compute an end result with 1 Hz. In this case we can ensure that each expert has classified and sent their results to the sensor fusion BeBot.

5. EVALUATION

In this section we show results of the evaluation of our experts. It is mainly about the reliability and quality of the data that the different classifiers compute. The basic setup to evaluate the expert programs was similar to the setup shown in Figure 1 in chapter 2. The setup was static and the robots were standing around the object in a distance from 10 cm to 20 cm.

5.1. Microphone

We tested the microphone classifier for reliability in four scenarios: silence, scream, music and simultaneous music and scream. For silence there was no sound source around the BeBot, for scream we shouted at the BeBot, for music we placed a sound source which plays music next to the BeBot and for music and scream we combined the scream and music test into one test. In Table 1 you can see the feature values for capturing the sound via “arecord”. We used the features as input for our classifier. The values of

silence	scream	music	music + scream
0.00030	0.05374	0.00852	0.15363
0.00030	0.16327	0.01097	0.52365
0.00030	0.12654		0.51002
0.00030	0.14014		0.14742
0.00030			0.46489
0.00030			0.33581
	0.00030	0.04685	
	0.00042	0.03314	
		0.05246	
		0.06689	

Table 1: Shows the value of our feature for captured sound via “arecord”. Values above the line are classified into correct classes and values below are wrong classified.

silence test are constant low as expected and right classified as silence. The values scream test are classified as scream, but some values are so deflected that they get classified as silence. The failure classification depends on using a little snippet of the sound file. As we mentioned we capture a one second sound file. For the classification we use only the beginning of this file. Due to the rest of sound we are losing many information for classifying it correctly.

Test series with music produces values that are classified to the “scream” class. Captured music creates frequencies in the range of our criteria for scream which is human voice. To differentiate between these classes we need to specify our frequencies in our classifier. Simultaneous playing of music and scream tests creates the highest value in our tests.

Due to the music and the screaming the values results from many frequencies in our range of human voice.

We compared our chosen frequencies with another possible range where formants are. Formants are spectral peaks of the sound spectrum of the human voice. On the one hand we have the range from 250 Hz to 2000 Hz where typical frequencies of human voice appear and on the other hand the range of 320 Hz to 1000 Hz, where the first formant for all vocals appear. In principle the smaller range of frequencies does not optimize the feature for our classifier. The second formants seem to be very important for our feature. Because of that we need the range of 1000 Hz to 2000 Hz for a better classification.

5.2. Camera

Due to the fact that the camera expert is divided in two different experts, we will evaluate these experts separated.

5.2.1. Face Detection

At first, the accuracy and the computing time of the learning and classification steps of the LBPH algorithm will be shown. Afterwards we take a look on Eigenfaces.

We tested our face detection with two different persons in a distance of 20 cm from the camera. The results of the LBPH are shown in Figure 8. With an average classification value of 64,53 % of Person A and 44,97 % of Person B, the classification results of LBPH are not very clearly. The results are also very distributed, so the average deviation of Person A is 16,21 % and of Person B 8,14 %. The average computing time in 84 images is 2136 ms per classification. The training of the classifier is done in about 4000 ms.

The Eigenfaces show other results (Figure 8). Here we have an average classification value of 93,3 % of Person A and 77,79 % of Person B. Further these classification results are not as distributed, so the average deviation of Person A is 7,17 %, the average deviation of Person B is 6,15 %. Eigenfaces need an average computing time of 113 ms per classification measured over 470 images. About 10.000 ms are needed to train this classifier.

We have also tested the Eigenfaces to false positives. In 30 classifications of different objects we get a maximum value of 42,2 % as face classification result, the average is 30,56 %. In 37 images with no special object in front of the camera we get a maximum value of 19,26 %.

We have also tested the Eigenface classification in different distances between the face and the camera. The classification result decreases by increasing the distance. Up to 30 cm, the results are almost 100 %. Then the results decrease. At a distance of 50 cm, the classification result is about 60 %. At the distance of 100 cm the values have fallen to about 35 %, which is still a deflection compared

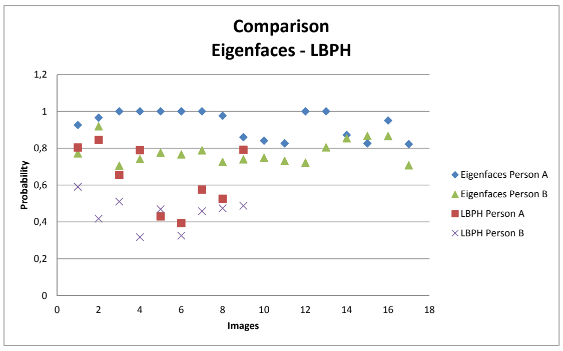


Figure 8: Comparison of Eigenfaces and LBPH

to the default values with no special object in front of the camera.

5.2.2. Motion Detection

The motion detection is a quite robust component. If there is no motion in front of the camera, the motion detection has never calculated a classification value above 5 %. Figure 9 shows the classification results when a person is crossing the camera. The motion detection computes one frame in an average time of 115 ms measured over 800 classifications.

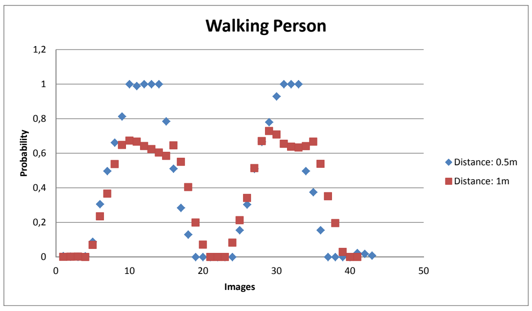


Figure 9: Classification results of a person crossing the camera perspective. The person enters the camera perspective at image number 5 and first leaves the perspective at around number 20, depending on how far the person was away from the camera. At image number 25 the person re-enters the perspective and leaves at image number 38 (.5 m distance) or image number 40 (1 m distance).

5.2.3. Parallel Computing of Motion and Face Detection

As mentioned in 4.2 it is possible to have a face detection and motion detection in parallel. This parallel computing changes the computing time. So the motion detection subside from 115 ms per frame to 123 ms, measured over 200 classifications if Eigenfaces is running in parallel, and 127 ms over 200 classifications if LBPH is running in parallel.

If Eigenfaces and the motion detection are running simultaneously, the computing time of Eigenfaces subsides from 113 ms per frame to 118 ms per frame, measured over 134 classifications. If LBPH and the motion detection are running simultaneously, the computing time of LBPH subsides from 2136 ms to 2206 ms measured over 80 classifications.

5.3. Infrared

To evaluate the infrared classifier we used different tools to create ir-waves. We used an electric blanket, a hot air gun and a light bulb. The tests with the hot air gun and the electric blanket failed, due to the same reason we can not detect humans with the ir-sensors. The reason is, also explained in chapter 4.3, that the sensors have a limited spectrum of infrared waves they can sense and humans, electric blankets or a hot air gun do not emit infrared waves in that special spectrum. The light bulb does emit infrared waves in a big spectrum and that is why the sensors can measure them and we can classify them. We assume that a fire in a rescue scenario will also emit ir-waves in a large spectrum and our classifier therefore can compute reasonable data.

distance	probability
0m to 1m	$> 90\%$
1m to 1.5m	$>70\%$
1.5m to 1.75m	$> 50\%$
1.75m to 2m	$< 50\%$
2m and more	$< 22\%$

Table 2: Classification results of a light bulb according to distance

In our test case with the light bulb in a lab, it was possible to classify the light source about a distance of two meters. Table 2 shows detailed results according to the percentage the classifier computes and the distance between the robot and the light source.

5.4. Sensor Fusion

To evaluate the sensor fusion component of our system we had to record data with the whole expert system running. We setup four different scenarios to record data that are relevant for the task of finding victims or fire with the multi sensor system. The first one is a scenario where the BeBots find a human lying on the ground asking for help from time to time. In the second scenario the system had to classify a light bulb, which simulates fire. The third scenario where no human or light bulb take place is for classifying “noise”. And than we had a last scenario where we had fire and a human. In this case the priority was to detect the human that has to be rescued. In all scenarios a static setup was used where the BeBots did not move.

We used the Wekatool [6] to evaluate our recorded data with a Naive Bayesian maximum a posterior classifier similar to our implemented classifier and a Multi Layer Perceptron [10]. For the training of the classifiers we split the data with tenfold-cross-validation rule.

classified as $\rightarrow$	human	fire	noise
human	181	5	7
fire	9	123	5
noise	11	2	50

Table 3: Confusion matrix Bayes approach

The Naive Bayesian maximum a posterior classifier, short MAP classifier correctly classified 90 % of the instances. The confusion matrix in Table 3 shows the results in more detail. On the diagonal one can see the correctly classified number of instances, the rest of the matrix are wrong classifications.

classified as $\rightarrow$	human	fire	noise
human	180	5	8
fire	4	132	1
noise	8	0	55

Table 4: Confusion matrix MLP

The MLP classified 93 % of the instances correctly which is a improvement of 30 % compared to the Naive Bayes classifier. But considering Table 4 compared to Table 3, one can see that the classification of “noise” with the MAP classifier has 38.5 % more wrongs then the MLP. Pretty similar behavior of the classifiers with the class “fire” where the MAP has 74.3 % more wrong classified instances then the MLP. Only at the class “human” the MAP could do 12 % better then the MLP.

6. DISCUSSION

In this work we build a distributed multi sensor system with the robot BeBot. The classification value computed in this system is able to classify both human and fire and this could be a help to human rescuers in a hazardous disaster scenario. But we have to face up our system in the context of a real disaster scenario, because our tests and evaluations took place in laboratories and not in the real world.

6.1. Lab results

The laboratory results show that our system is running quite robust in test scenarios, as mentioned in section 5.

6.1.1. Microphone

For classifiying via microphone we had the goal to recognize if there is human voice. The classifier we present in this paper is in principle able to identify human voice. In general we had problems with noise in capturing part. To specialize the classification of human a detection of formants could be used to have more specific frequencies of human voice.

6.1.2. Camera

The algorithms Eigenfaces and LBPH yield very different results. Comparing both classifiers show that both Person A and Person B can be better recognized by Eigenfaces than LBPH (Figure 8). This combined with the fact, that Eigenfaces are much faster in the calculation of one frame (Eigenfaces: 113 ms, LBPH: 2136 ms) led us to use the Eigenfaces for the face detection.

6.1.3. Infra Red

The classification with a Gaussian estimator and an exponential estimator so far works fine, compare to chapter 5.3. But the infrared expert lack the classification of the class "human". This is an important task in a rescue scenario and can be accomplished with infrared sensors that have a greater spectrum of infrared waves then the BeBots sensors have.

6.1.4. Fusion

The results of chapter 5.4 show, that the sensor fusion with the Naive Bayesian approach has some disadvantages compared to a neuronal network like a Multi Layer Perceptron. The MLP in general classifies better, because the Perceptron does 30 % less misclassification's. This is why the implementation of a Multi Layer Perceptron or more general a neuronal network would pay of.

6.2. Real World

The distributed multi sensor system in this work, was able to distinguish between human victims and dangerous areas where fire burns. The BeBots served well as platform to build the classifier system and to evaluate the system on the robots. In the context of a real disaster scenario, one should use a different operating platform then the BeBot, because it was designed for teaching and student work in a lab. Due to the flexibility that comes through the use of small components communicating via RSB, it is possible to change the platform very fast and easily.

A different point is the use of a distributed multi sensor system compared to a multi sensor system that runs on one bigger robot. A disadvantage of the distributed system is, that we need multiple robots in front of an object to classify it with a high reliability with data from different sensors. Advantages of the distributed system on the other hand are surely the cost efficiency of many cheap robots, the fact that they can cover more ground in a searching scenario and that they are replaceable. In the case one robot fails the system would still be able to complete the task.

Looking towards future projects it would be interesting to build a "semi-distributed system". The idea is to combine the advantages of the one robot solution and our system. The key to that goal is an intelligent fusion program, that searches for objects with a simple sensor. When there is something detected by the robot it will start the other expert programs in a chain in order to have a classification with data from all sensors, in order to get a good reliability of the results.

7. CONCLUSION

We have successfully implemented the idea of a distributed expert system which was able to classify both human as well as a fire simulation with the combination and fusion of the classification of different sensor data. The modular hierarchy with the sensor behavior separation combined with the adaptive communication interface lead to a decentralized system, in which further sensors or further robots can be integrated very fast and easily.

8. REFERENCES

[1] Timo Ahonen, Abdenour Hadid, and Matti Pietikäinen. Face recognition with local binary patterns. In *Computer vision-eccv 2004*, pages 469–481. Springer, 2004.

[2] R.N. Bracewell. *The Fourier Transform and Its Applications*. Electrical engineering series. McGraw-Hill Higher Education, 2000.

[3] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer vision with the OpenCV library.* " O'Reilly Media, Inc.", 2008.

[4] Micael S. Couceiro, David Portugal, and Rui P. Rocha. A collective robotic architecture in search and rescue scenarios. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, SAC '13, pages 64–69, New York, NY, USA, 2013. ACM.

[5] Devendra P Garg and Manish Kumar. Sensor modeling and multi-sensor data fusion. Technical report, DTIC Document, 2005.

[6] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18, November 2009.

[7] Stefan Herbrechtsmeier, Ulrich Rückert, and Joaquin Sitte. Amiro – autonomous mini robot for research and education. Advances in Autonomous Mini Robots, pages 101–112. Springer, 2012.

[8] Stefan Herbrechtsmeier, Ulf Witkowski, and Ulrich Rückert. Bebot: A modular mobile miniature robot platform supporting hardware reconfiguration and multi-standard communication. pages 346–356. Springer, 2009.

[9] Yugang Liu and Goldie Nejat. Robotic urban search and rescue: A survey from the control perspective. *Journal of Intelligent and Robotic Systems*, 72(2):147–165, 2013.

[10] Fionn Murtagh. Multilayer perceptrons for classification and regression. *Neurocomputing*, 2(56):183 – 197, 1991.

[11] The University of Sheffield. Image engineering laboratory - face database. CTAN: `http://www.sheffield.ac.uk/eee/research/iel/research/face`, June 2014.

[12] William H Press. *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press, 2007.

[13] Dipl.-Ing. Dirk Fischer Prof. Dr.-Ing. Brbel Mertsching. Get lab. http://getwww.uni-paderborn.de/research/getbot, 2014.

[14] Alsa Team. Alsa-utils, http://www.alsa-project.org/main/index.php/download.

[15] Matthew Turk and Alex Pentland. Eigenfaces for recognition. *Journal of cognitive neuroscience*, 3(1):71–86, 1991.

[16] Johannes Wienke and Sebastian Wrede. A middleware for collaborative research in experimental robotics. In *IEEE/SICE International Symposium on System Integration (SII2011)*, Kyoto, Japan, 2011. IEEE, IEEE.

INTELLIGENT SYSTEMS PROJECT: THE NEXT GENERATION GUI

S. Khan, R. Indoria, L. Ruegemer

Faculty of Technology, Bielefeld University
Bielefeld, Germany

Supervisors: H. van Welbergen, P. Kulms

ABSTRACT

Many advanced Human/Computer interaction devices, such as speech or face recognition, virtual agents or graphical interfaces are available but experiments and research often only concentrate on one system. As the creation of a user interface using multiple state-of-the-art input and output devices, and managing the interaction of these devices is hard, we propose a method to design these interactions using simple state charts to make it feasible for non-specialists to create user interfaces that include multiple different devices to experiment and prototype with next generation interfaces. We conducted a use case study to assess the system and its user friendliness.

1. INTRODUCTION

Technology is evolutionary, it keeps changing. The same goes with user interfaces. The earliest type of interface was called “Command line interface”, where a keyboard was used to access a program in a computer using a set of commands. With the addition of pointing devices (such as the computer mouse) the interfaces evolved to “Graphical User Interfaces”. Nowadays the inputs are no longer restricted to mouse and keyboard but also includes other pointing devices such as a stylus, joystick, and touch screen. The computer device can be mobile, tablet, PDA, MP3 player and even TV.

Now user try to design “next generation user interfaces”. They have several heterogeneous input and output devices available. Integration of these into a single interface is difficult especially for non-programmers or researchers of different institutes as the original developers. This results in research that is concentrated on only one or few of these new technologies.

Frameworks like Crosstalk [1] use state machines to script the scene flow with virtual agents. As state machines provide easily understandable way to create content even for creative, non-programming users. We want to expand on the idea of using state machines to make it feasible designing a next generation user interface and controlling the interaction between different systems.

2. SYSTEM DESIGN

Figure 1 shows the users concept and the three important parts of next generation interface design. The main objective of this project is to create a framework that makes it possible to easily integrate different devices into one interface or program. As interfaces are event driven, it is possible to describe the behaviour with finite state machines, so that it should be possible to design the interaction using State Chart Extensible Mark-up Language (SCXML). As the potency of the different technologies like virtual agents rises the framework should be easily expandable to handle different devices. Designing the graphical interface is a big part so we want to use an established software to make the creation of these as easy as possible without developing a new interface designer.

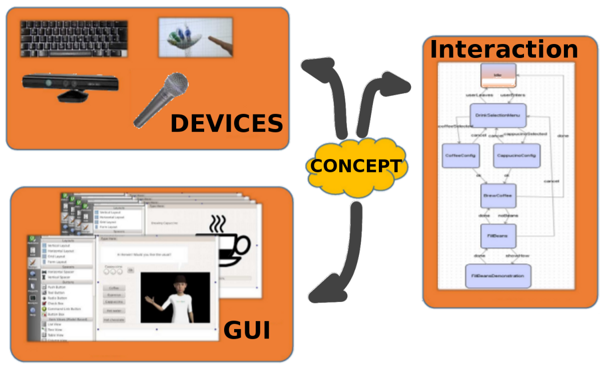


Figure 1: Three parts of next generation user interfaces

3. SOFTWARE COMPONENTS

The used software components include:

1. Apache commons SCXML which is used for implementation of an engine aimed at execution of a program behaviour defined by a finite state machine as SCXML file.

2. The Ipaaca middle-ware that is used to realize the communication between the SCXML engine and external programs. Ipaaca implementations for Java, Python and C++ are available. Ipaaca uses incremental units [2] containing one or more strings to sent messages from an output buffer in one program to the input buffer of another program.

3. The Articulated Social Agents Platform (AsapRealizer) which is the Behaviour Markup Language (BML)[3] realizer for incremental, fluent and multimodal interaction with a virtual human or robot. BML is used for defining gestures, facial expression and speech performed by a virtual agent.

4. Qt4 is used as a cross-platform interface realizer. The Qt designer is used to generate UI files.

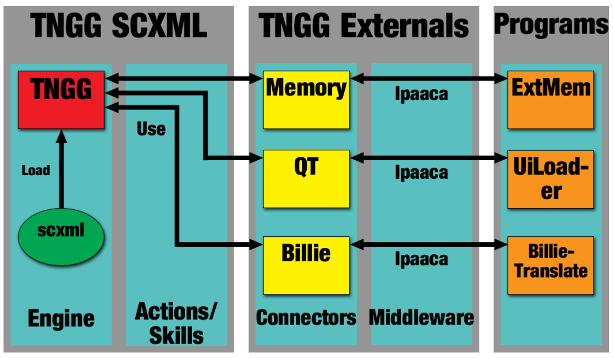


Figure 2: TNGG Architecture

The architecture of the TNGG engine is shown in Figure 2. TNGG(SCXML) section 3.1 loads the SCXML file, executes the correct skills and actions and reacts on events to control the program flow. Skills and actions use features of TNGG(Externals) section 3.2 which handle the communication with other systems. An Example would be BillieConnector which provides a say function to let the Billie agent speak. Externals can be used to hook-up different devices, currently connectors for a memory, qtinterface and the billie agent are implemented. The connector are responsible to start the needed software, in the example of the BillieConnector the Agent itself, the ASAP realizer[4] and the BillieTranslator. The developed software components will be briefly explained in the following subsections.

3.1. TNGG(SCXML)

The main module handles loading and execution of scxml files using Apache Commons SCXML[5]. To run user defined behaviour, an easy to use extension of apache commons AbstractAction was designed. These will be automatically linked during load of the scxml. Actions will be automatically executed by the apache commons SCXMLExecutor. The possibility to create functionality is expanded with TNGGSkills. These will be executed if the id of a state matches the name of the skill with a prefixed "$". In difference to actions, skills will automatically fire "skillname.success" or "skillname.failure" after execution.

3.2. TNGG(Externals)

To use and control the interaction of different systems with TNGG, the communication between them and TNGG is defined in this sub project. Currently the ipaaca middleware is used but the usage of others is possible. For each system used a "AbstractConnector" has to be implemented. The purpose of these connectors is to start the different systems on demand the installation prefix of external software can be changed with the "-psoa" and "-pext" arguments. Also ipaaca handlers on the specific per connector defined prefix(ipaaca category) are created. The default handler parses all messages received on "prefix" for "event" and "memkey". Events get prefixed and fired in the state machine . Handling of "memkey" is described in 3.2.2. The following subsections describes the developed "External Programs".

3.2.1. UiLoader

The UiLoader is a c++/qt program that creates and shows QWidgets by loading ui files during runtime. The action "ActionUiLoad" used the QtConnector to send a load message containing the filename of the ui file which triggers the creation of a gui window. Interact able objects get connected to send their object type and object name as events over ipaaca. Objects like combo boxes send their value on start and on every change over ipaaca so they can be put into the connected memory. The name of the Object is used as memory key.

3.2.2. ExtMem

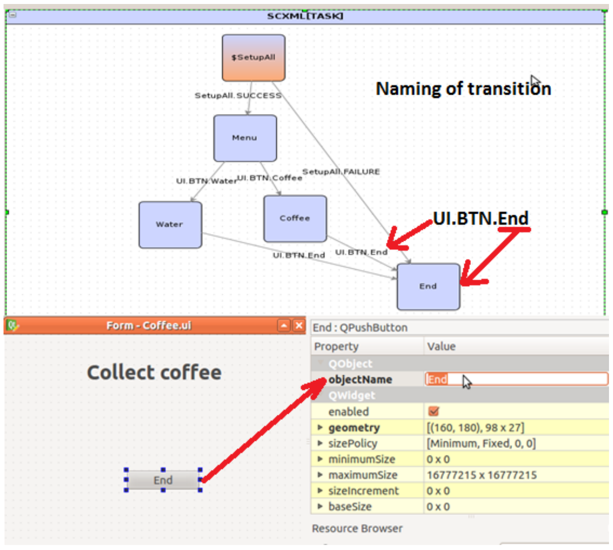
Simple implementation of a memory using key-value pairs to store information. If the MemoryConnector is initialized the ipaaca payloads of every connector get parsed for memory key-value pairs and saved into memory with "prefix.key" as key. Memory is read by creating a message that contains the key. This is then published over ipaaca and filled with the corresponding value from ExtMem. As soon as the value is written the MemoryConnector receives the change and return the memory values.

3.2.3. BillieTranslate

This External Program translates strings into BML[3] and send over ipaaca to the agent so they can be executed by the asap framework. The functionality of this program could be included in the connector but as the asap framework uses the ipaaca middleware, BillieTranslate is external to not create dependencies on the middleware inside TNGG.

4. INTERACTION/OPERATION EXAMPLES

Desiging a user interface interaction with example of drink selection state chart XML(SCXML) file Figure 3, shows the drinkselection menu SCXML file and below it is Coffee.ui user interface file linked to Coffee state in SCXML file. For



End : QPushButton

Property	Value
QObject	
objectName	End
QWidget	
enabled	✓
geometry	[(160, 180), 98 x 27]
sizePolicy	[Minimum, Fixed, 0, 0]
minimumSize	0 x 0
maximumSize	16777215 x 16777215
sizeIncrement	0 x 0
baseSize	0 x 0

Figure 3: State chart XML file for drink selection menu and coffee.ui user interface file

making the interaction with user interface file, 3 steps are involved:

4.1. Making User Interface file

User design the user interface using Qt Designer. User can make use of widgets like buttons, spinboxes, etc. to show in ui file. In example of figure 3 widgets push button is used. This button is given label as ”End” and also object name for push button is ”End” which is shown in lower right hand side in figure 3 pointed with arrow.

4.2. Making SCXML file

For creating SCXML file, we used fsm-editor to create SCXML file. fsm is finite state machine editor for making SCXML files. A corresponding state in SCXML file can be linked to user interface file. For use case study, we made SCXML with 5 states namely Menu, Coffee, Water, End state. First state $SetupAll was made to run skills in background to make system ready for execution.

4.3. Transition from user interface file within a SCXML file

For example, to make the transition between End and Coffee states in SCXML file, see figure 3, for triggering of events like pressing of button by user, can be implemented by using syntax ”UI.BTN.ObjectName”. Here, UI: represents the ui file, BTN: represents the button in ui file, ObjectName: represents the name of Object like push button in user interface file. In figure 3, we have used objectName as ”End”. So, for making transition from previous state to the state which link this user interface file we have to label transition between these two state as ”UI.BTN.End”.

SCXML actions were used for connecting user interface file and sending message to agent billie.

5. USE CASE STUDY AND RESULT

5.1. Method

The system evaluation objective was to check the user friendliness of the system as whole. Evaluation was required to reach our goal of making an easy to use system and to improve some minor bugs. Evaluation helped us to demonstrate our programs progress.

The main goal in use case study was to create a Graphical User Interface. The main task was divided into four subtasks, to check the progress in steps and give some crucial information about the system to the user. For every task completion, relevant information was provided through the documents.

The first task was to run the TNGG system with state chart XML(SCXML) file and get to know about working of the system. The main idea was: To make user familiar with SCXML and its use in our program. The user would learn about the states and events like pressing of button in user interface that triggers transition between states. The user were asked to look in SCXML file and answer some questions about the states at which they were present before each events.

The second task was to connect a user interface file with one state in SCXML file. The idea was: to make user understand about the actions and how to use these actions in SCXML language. The user had to learn to connect user interface file with particular action and how the user interface file will pop up whenever a particular state was reached.

The third task was to make a transition between two states through user interface file by editing the transition name. Here, the user would learn about: Naming of transition for receiving the events like clicking of button from user interface file.

The fourth task was to send a message to artificial agent Billie and make him speak that message. Through this task user gets to know: Which action to be used for sending messages and how to connect this action in one particular state in SCXML file. By completing this task, the user was able to make graphical user interface with artificial agent Billie.

We ran the use case study with eight participant. They all were technical user with computer science background. All the users were male and within the age group of 22-32 years.

We collected qualitative and quantitative data from user. Qualitative data: In this users, were handed evaluation sheet before case study and were asked to rate their programming skills and expertise with artificial agents. After user case study was completed user were asked to fill questionnaire about ease, understanding of task and clarity of documentation. Quantitative data: This data was collected in form of time required to perform each sub tasks and answer the set of question in each subtask.

5.2. Results

We are showing the results of each and every task in a box plot graphical representation and their explanation. In each, task we are showing the median, upper and quartiles, minimum and maximum data values. Boxplot graph is given below

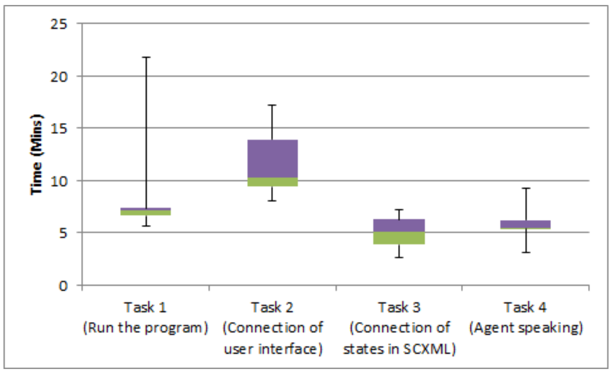


Figure 4: Box plot showing time completion for the individual tasks

Task 1: We identified the average time taken by the user to complete this task was 7 minutes 14 seconds. Here, users took more time to understand the state chart, but they got complete idea of the program through the changing of states in debug window and then they were able to answer the questions very easily. Some users, who were familiar with the SCXML, they performed this task very quick.

Task 2: Here, user took lots of time in understanding Action and then editing in namespace. The average time taken by the users was 10 minutes 30 seconds. This time result can be seen in figure 4.

Task 3: This task was to make the user familiar with the transition between two states through user interface file by editing the transition name in namespace. In this task, users took on an average 5 minutes 4 seconds. This shows after performing task 2 users were able to grasp fast about the transition naming and performed it pretty quickly.

Task 4: This task took on average 5 minutes 40 seconds. In this task, user learnt how to use action required to perform it from previous task.

Overall the user case study was completed by user on an average 28 minutes and 28 seconds. In this use case study, all the users were able to complete the task. However, two user could not answer the question at the end of task 3 about the naming of transition and its meaning as they didn‘t understand it. We conducted the user review about the usefulness, ease and clarity of documentation, understanding of task, ease in completion of task and their satisfaction. We ask user to rate from range 1 to 5 where 1 meaning not clear and 5 means very clear. The average rating for the ease of understanding documentation was about 4.0 out of 5.0 which mean it was clear with some minor confusions.The average rating of ”understanding of task” is 3.5 out of 5.0 which means it was clear to user why they are performing task. The average rating for ”ease in completion of task” is 3.5 out of 5.0 which means user were able perform the task without any big problem. Some users also gave useful suggestions about the documentation improvement like explanation in picture itself rather then describing in paragraph.

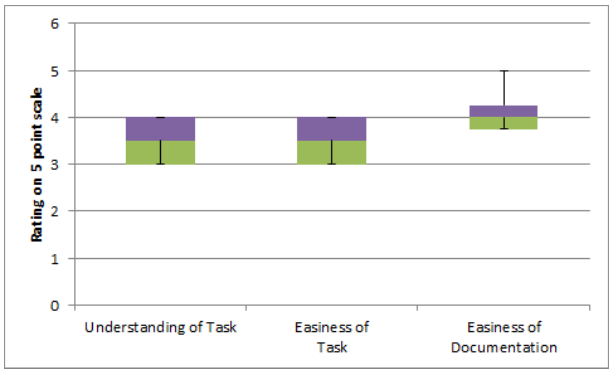


Figure 5: Box plot showing user ratings for task and documentation

6. FUTURE WORK

TNGG system design now has input from keyboard and mouse,and interaction with artificial agent Billie. The first thing to implement in future is to add some more input and output devices like camera, microphone,face detection, speech recognition system for voice interaction with user.

7. CONCLUSION

We have presented a methodology to connect different devices together for their intuitive interaction(UI elements), conversational form of virtual agent. We conducted use case study for the system and took user review about ease and friendliness of the system.The main outcome of the use case study was that user were able to make their GUI interface easily and were satisfied.

8. ACKNOWLEDGEMENTS

We would like to thank our mentors for there guidance and persistent help for this paper. In addition, we would like to say special thanks to participants of our study, who helped us in evaluation of user study and finding the best result of our project.

9. REFERENCES

[1] P. Gebhard, M. Klesen, and T. Rist, "T.: Authoring scenes for adaptive, interactive performances," in *In: Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*, 2003, pp. 725–732.

[2] D. Schlangen and G. Skantze, "A general, abstract model of incremental dialogue processing," *Dialogue and Discourse*, vol. 2, no. 1, pp. 83–111, 2011.

[3] ASAP-Project. (2014) Behavor markup language. [Online]. Available: http://asap-project.ewi.utwente.nl/wiki/BML

[4] S. Kopp, H. van Welbergen, R. Yaghoubzadeh, and H. Buschmeier, "An architecture for fluid real-time conversational agents: integrating incremental output generation and input processing," *Journal on Multimodal User Interfaces*, 2013, online First Article.

[5] Apache Software Foundation. (2014) Apache commons scxml. [Online]. Available: http://commons.apache.org/proper/commons-scxml/

[6] QT Project Developer. (2014) Qt4. [Online]. Available: http://qt-project.org/