Controlling Smart Environments using a Brain Computer Interface

Gernot Ruscher* Frank Krüger Sebastian Bader Thomas Kirste

Universität Rostock,
Albert-Einstein-Str. 21,
18059 Rostock, Germany

*corresponding author: gernot.ruscher@uni-rostock.de

ABSTRACT

We describe first experiments for controlling smart environments using a brain-computer interface. The graphical user interface is automatically synthesised from device models that specify effects of device functions on the environment. Thus, the number of interactions can be reduced, and a novel way of human machine interaction is introduced: Controlling the environment instead of single devices.

Categories and Subject Descriptors

H.5 [**User Interfaces**]: Graphical User Interfaces, Input Devices and Strategies

General Terms

Design

Keywords

BCI, smart environments, graphical user interfaces

1. INTRODUCTION & MOTIVATION

Brain Computer Interfaces (BCI) are ongoing research since the 1970s [10], employing invasive technologies as well as non-invasive approaches such as EEG. Key potential of BCI is the possibility of man-machine interaction without requiring motor activities: Hands free, no gestures, no speech, no pointing and clicking. Recently, low-cost devices have become available at market targeting the gaming scene, claiming, at a very competitive price, to provide the capability of cerebral control for at least a limited set of interactions.

Our experience so far shows that with those simple BCI devices, interaction is kept within tight bounds due to the limits of this communication channel: A merely small character set is available at a low frequency, which leads to a severely limited data rate. Applications based on low-cost BCIs thus have to deal with these limitations and to adapt their graphical user interfaces (GUIs). These need to optimise the number of user interactions necessary to trigger an intended application function. Thus, highly application-specific GUIs need to be implemented, and various approaches have been developed that are aimed at grouping functions *smartly*.

With the vision of Ubiquitous Computing coming true, device become more and more *invisible* to the user, and hence cause the need for novel user interfaces. One specific application field in this context is that of *smart environments* [7]. These build complex sets of heterogenous devices, partly fixed to the environment and partly brought-in by the user. Thus, applications in smart environments need to base on such a dynamic ensemble of devices which are possibly unknown in advance. Developing user interfaces which provide control options for lots of devices with lots of different functions would be a tough task by itself. In addition with the dynamics of the underlying device ensemble it quickly seems to be insolvable.

One approach to provide a user interface for a dynamic device ensemble would be the synthesis of a dynamic GUI from formal device descriptions. Various approaches, for example built upon UPnP [4] or Jini [1], make it possible to generate GUIs, even for the control of a dynamic device ensemble. Unfortunately, users' experience shows difficulties with these approaches.

Our approach presented in this work relies on the following working hypothesis: What users of a smart environment are interested in is not the individual device, but their effect on the environment. One simple example: When a user switches a lamp on, he actually just wants to increase the lightness of the room. In this way, all the lamps of this room are able to increase the lightness and would therefore be redundant with respect to their effects on the environment.

With this article, we describe first experiments to control a smart environment using the neural impulse actuator (NIA), a low-cost brain non-invasive computer interface. We use semantic models of the environment and the devices. We model the devices with respect to their specific influence onto the environment. We present a principle approach for the synthesis of graphical user interfaces in order to reduce the number of necessary interactions.

SEMAIS'11, Feb 13 2011, Palo Alto, CA, USA

2. PRELIMINARIES

After presenting the neural impulse actuator, we briefly discuss our lab used for the experiments. Furthermore, we give a short introduction to STRIPS: A formalism to describe preconditions and effects of operations.

2.1 The Neural Impulse Actuator

In 2007, OCZ Technology Group Inc. [5] introduced the *Neural Impulse Actuator* (NIA): A simple BCI controller, basically a headband, equipped with three electrodes capturing electrical potentials from the forehead. Those potentials include electromyogram (potentials arising from muscle control), electroencephalogram (signals from the nerves in the brain) and electrooculogram (signals coming up during eye movement). A special controller is used to connect the sensors to the computer. The NIA registers itself as a USB human interface device, which basically permits it to act like any other input device, e.g. a keyboard or mouse. Figure 1 shows a photo of the NIA controller.

After calibrating the NIA, it is supposed to be usable as a virtual joystick and to switch events, which can be triggered by different electric potentials or muscle movements. In our experiments, we found the following actions easy and stable to recognise:

- eye movement in general,
- heavy muscle movement on the forehead, or moving the jaw,
- light muscle movement on the forehead,
- heavy thinking, and
- relaxing, or closing the eyes.

Here, we want to use the NIA to control our lab environment, even while working on other subjects. Hence, inputs triggered by heavy thinking and relaxing are not suitable signals for a smart environment controller. However, parallel performance to compose more complex signals does not seem to be helpful, as we want to provide an easy-to-use interface. Therefore, we have at most three distinguishable signals at hand: (i) eye movement, together with (ii) heavy and (iii) light forehead muscle movement. To complicate things further, users of the NIA can perform those signals at a merely low frequency of about 10 per minute at most, leading to a comparatively low data rate.

2.2 Our SmartLab

For our experiments we utilised our SmartLab: An instrumented meeting room (cf. fig. 2) equipped with a number of remotely controllable devices. It is frequently used as a

Figure 1: The NIA

Figure 2: Our SmartLab.

room for lectures, presentations, and meetings, but also as an experimental setup for user studies.

Our lab is equipped with a couple of sensors, needed to observe state changes in the room. There are e.g. sensors capable of detecting whether the windows are closed or opened, measuring the current temperature, or detecting persons that enter or leave the room, and estimating their number and current positions [3].

On the other hand there is a number of remotely controllable devices required in typical meeting rooms: Dimmable lamps as well as movable projection screens and sun shades, controllable via EIB [2], a computer video and audio matrix switcher to connect brought-in devices with the installed projectors and audio equipment, just to mention the most important. Those devices are actuators in essence, but can also be seen as specific sensors, in each case providing access to their respective status.

Our lab features a powerful middleware (as for instance described in [6]) which on the one hand allows for control of all existing hardware using simple commands. On the other, besides triggering device actions, our middleware enables every device to make its specific properties accessible to other components in the system.

2.3 STRIPS

To describe the capabilities of devices and their possible actions, we suggest to use STRIPS-operators as illustrated in [9]. Those operators formalise (i) *preconditions* that need to hold to make the execution of the respective operation possible and (ii) *effects* that specify the world state changes provoked by the execution of the respective operation. STRIPS-operators have successfully been used in the context of smart environments before [8].

Due to their associated declarative semantics, they are well suited for an automatic interpretation and hence for the construction of a controller. We annotate every operator with the middleware command which needs to be executed to perform the operation. Figure 3 shows two simple operators describing how to switch a lamp `l` on and off.

SEMAIS'11, Feb 13 2011, Palo Alto, CA, USA

3. OUR APPROACH

As depicted above our user interface needs to cope with a dynamic ensemble of heterogenous devices, which is the reason we do not have the option to hard-code a certain controller for a fixed environment. Therefore, we need to consider ways and means of synthesising a controller from an abstract description of the environment and the devices.

According to figure 4, we consider three different modelling tiers: Top left in the sketch is the layer of the *background model*. It specifies existing parameters of the environment and how they can be modified. To provide an example we have formalised two specific parameters and their respective operations in figure 5: There exists some parameter that corresponds to the *lightness* of our room, and it can be modified by two operations named *increase* and *decrease*. The parameter *temperature* is handled likewise. This explicit kind of formal specification of environmental parameters is needed later on to describe the effects on the environment caused by triggering device actions.

As illustrated in section 2.2 all of our devices are made accessible through our middleware, that way providing on the one hand the entire set of current properties to other components and on the other an easy-to-use interface for triggering device actions. In figure 4 this device representation layer is called *device model* and establishes a certain level of abstraction from the plain hardware, where every device can exist without requiring local knowledge on the existence of other devices, the middleware or even the environment. Besides properties and actions this layer holds additional information on the device such as the device type, its name, or whether the device is currently available in the system.

The third modelling tier, the *effect model*, now etablishes the relation between the raw device descriptions from the device model and the environmental parameters from the background model. This is done by modelling device actions with respect to their effect on the environment. Every action is annotated with a formal description of the influence of its execution on the specified environmental parameters. As depicted in figure 3 for instance the execution of the `turnOn` method of a lamp has an effect on the environmental parameter lightness in the form that the latter would be increased. We suggest to use STRIPS as modelling formalism to describe the semantic meaning of actions to the environment.

ACTION(switchOn(l,r))
    PRECOND: Lamp(l) ∧ Room(r) ∧ In(r,l) ∧ Off(l)
    EFFECT: ¬Off(l) ∧ On(l) ∧ Lightness.increase(r)
    COMMAND: `l.turnOn()`
ACTION(switchOff(l,r))
    PRECOND: Lamp(l) ∧ Room(r) ∧ In(r,l) ∧ On(l)
    EFFECT: ¬On(l) ∧ Off(l) ∧ Lightness.decrease(r)
    COMMAND: `l.turnOff()`

Figure 3: An annotated STRIPS-operator describing the switchOn and switchOff actions for a lamp. Every operator contains preconditions and effects of the action, and the command to be executed to perform the action.

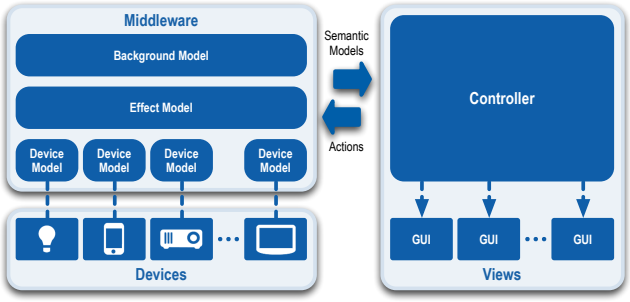

Figure 4: Integration outline of the proposed controller within the smart environment system.

The information aggregation mechanisms of our middleware enable applications to gather these effect models analogously to the previously mentioned properties as well as type and status information of devices. Therefore a GUI application – called *controller* in figure 4 – is now able to provide itself with a list of all devices together with their descriptions, i.e. the type (lamp, sun shades, ...) of the device and all its available actions, including their particular preconditions and effects. After collecting the operators, it can generate diverse GUIs views. Below we present some first experimental views that shall demonstrate the descriptive power of our proposed approach.

As mentioned above, there are basically three different actions (keystrokes) a human can reliably perform. Based on the formal description of our devices we implemented lab controllers tailored for a limited communication between human and computer to evaluate our approach. We designed them following the Mac Finder's *Column View*. Two keys are used to move the focus up and down a list, the third to select the item.

Our very first prototype contained three columns, of which the first contained a list of device types, the second all available devices of the type selected in the first column, and the third all the applicable functions of the device in the second column. This prototype does not involve any environmental knowledge yet. Due to its three-button-based design, this controller interface would enable users who have to rely on a NIA to take control of a complex and dynamic set of heterogeneous devices once these have been seen by the system through the middleware. But without further tweaking and tuning – which we elaborate on in section 4 – the menus would be very large if they contain every possible action for every possible device.

PARAMETER(Room, Lightness)
    OPERATION(Lightness, increase)
    OPERATION(Lightness, decrease)
PARAMETER(Room, Temperature)
    OPERATION(Temperature, increase)
    OPERATION(Temperature, decrease)

Figure 5: Background knowledge: Two environmental parameters and their respective operations.

SEMAIS'11, Feb 13 2011, Palo Alto, CA, USA

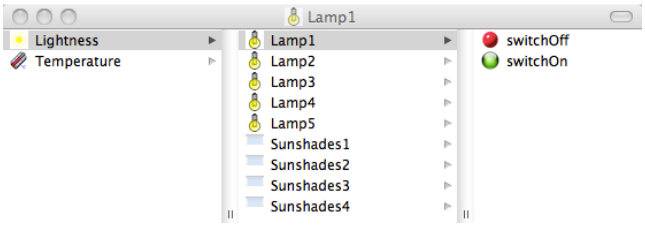

Figure 6: A user interface synthesised by our second controller prototype.

Our second prototype now extracts advantages from our formalisation: As an alternative to a static arrangement based on the device type, we can group our devices with respect to their effects on the environment, as we did in our second controller, which is shown in figure 6. Here, the first column has been replaced by a column containing controllable parameters as for example the lightness, or the temperature. Then, all those devices are listed in the second column which can actually influence the selected parameter. Finally, the third column would again contain performable actions.

But our formalisation of effects has further advantages: The previously depicted controller still displays all the devices even if they have similar influence on the environment. This leads to a long list of devices with each of them still providing every possible action. But, with respect to their effects they are kind of redundant and if we assume that a user is not interested in the device and its particular action itself but is interested in its effect, we can omit all these information and simply provide control options of an environment. For this purpose, we can simply use our existing model of environmental parameters. Our third controller prototype working this way is depicted in figure 7. In the first column it displays the environmental parameters of the background model, which can then be adjusted by selecting one of the items in the second column.

Within our research project MAike, all devices send their descriptions to a central look-up, realised as a tuple space [6]. Furthermore, the NIA controller integrates itself as a new modality for user interaction among other existing ones (speech interaction, intention analysis, ...). So far, we integrated four different device types, with at most eight devices and five actions, and initial experiments showed that those devices are easily controllable using this simple controller together with the NIA.

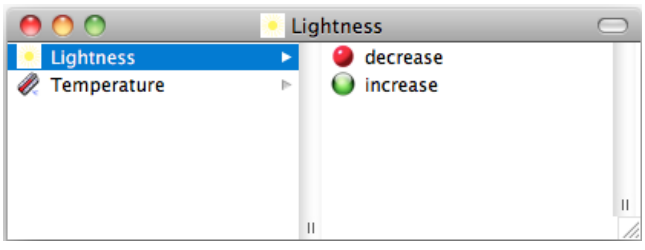

Figure 7: Another user interface synthesised by our third controller prototype.

4. CONCLUSIONS AND DISCUSSION

Our investigations leave us with mixed feelings. On the one hand, it is indeed possible to evoke a limited set of actions using the NIA controller. On the other hand, the concentration required from the user (and, indeed, the level of self control regarding the facial expression), in our opinion leaves ample room for optimisation of both signal acquisition at the sensory level and signal processing at the algorithmic level.

While developing the controller, we could reliably distinguish three different inputs only. Of course the NIA itself provides a richer interface in form of analogue joysticks, but we found that those are hard to control while concentrating on other tasks and they cannot be distinguished as reliable as needed to control the environment. Nonetheless, a better recognition of crisp events would be desirable for the future.

With this work we present a mechanism that deals with the issue of operating a complex and dynamic system through such restricted channels. If we had to deal with an environment that would be static and consist of a fixed number of well-known devices, we would be able to build a controller being perfect in terms of the number of interactions. Because we need an automatically synthesised controller, our approach uses STRIPS operators to apply semantic knowledge of the system's devices and their possible actions aiming at design (and synthesis) of adequate user interfaces.

In the simple approaches presented above, the menus would still grow very large with every new device entering the scene if the menus still contain every possible action for every possible device. Therefore, it is desirable to show the user not the whole menu. Instead, we should offer more abstract actions. Furthermore, we would like those abstract actions to be context dependent and automatically generated.

The focus of this work was not to develop the perfect-working graphical user interface for controlling tasks using the NIA controller. Our goal was to investigate several approaches to reduce the number of interactions a user has to perform to have a function executed. Therefore, the synthesised GUIs will never be the best ones one would find through manual design. This work considers ways and means of involving formal descriptions of environmental knowledge into automatic GUI development. We have not yet evaluated which are best suited for the present application case and different approaches from the ones depicted in this work are imaginable.

As mentioned above, the menu structure used in our prototype, does probably not allow to control larger environments. In the following section 5, we have discussed a number of methods to create more suitable menus and higher level actions. Those have to be implemented and tested with respect to their usability.

Nevertheless, our solution seems to be a principle approach for the synthesis of graphical user interfaces in general, which could bring a significant benefit not only for motorically challenged users, or those who require hands-free interaction in situations, where speech control is not an option.

5. A ROADMAP FOR THE FUTURE

The controller described above is applicable for small well-defined scenarios, but certainly not to control a complex infrastructure with hundreds of device-actions. This is due to the realised menu-based interaction. Considering a usual living room, it is not hard to think of many different devices which should be controllable. Just think of all lamps, shades, multi-media devices, etc. Therefore, alternatives are currently under investigation. We try to learn user preferences online and group devices dynamically. For example, we could group devices together as one *virtual* device if they have been used in one go a couple of times. Another grouping approach could be a Huffman-like encoding which would put the more important devices above others in the list.

As mentioned above lots of user evaluation needs to be done in order to find more intuitive user interface design methods that make use of formal descriptions of the effects caused by device actions on the environment and of the environment itself. One approach could be another control metaphor which is especially designed for brain computer interfaces: A rectangle that periodically rolls over a list of items on the screen and each time marks the currently covered item. The user now simply *thinks* of the particular item he wants to choose and a signal from the brain indicates the moment when the bar covers this item. This selection mechanism can not only be utilised with lists of items, but also like shown in figure 8: First a horizontal bar goes the up-down direction and stops when the height of the selected item is reached. Second, the vertical bar starts moving and is used to indicate the particular device in this line.

The area of automatic *intention analysis*, that is the detection of the user's goals based on his current activities, opens further possibilities. Given the user's current goals, we could automatically re-arrange the menus to provide faster access to device which are most likely to be used in the current situation. Going one step further, high-level actions could also be added to the controller. A room detecting the start of a lecture could offer the compound action like "put my presentation on the projector", which consists of several smaller actions. For this purpose some additional *strategy synthesis* component would be needed.

We have not yet fully covered the possibility of the existence of multiple environments, containing different devices, but controllable through the same user interface. This could be the case in a smart home consisting of several smart rooms, where users possibly would like to have functions performed in other rooms remotely or trigger actions in different rooms simultanously. One straightforward approach would be an additional column inserted before the first one, that specifies the particular room. After this column would be the ones depicted in the previous sections.

So far we have been relying on the feedback on an existing computer screen. This is possible for users that can carry a small display with them. E.g., for people depending on a wheel chair, this monitor can be integrated into it. Alternatives should be investigated, e.g. sounds or small displays right next to the devices which are controllable. This would enable the user to move freely around in the environment without watching a computer screen for every action.

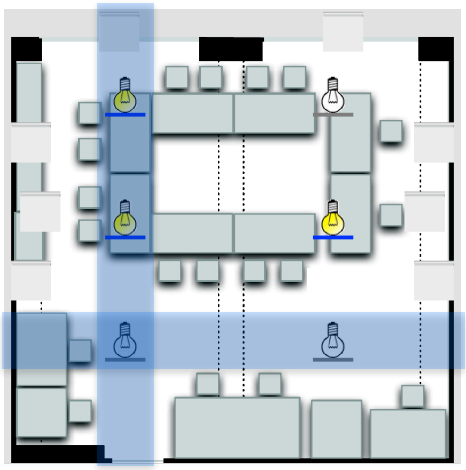

Figure 8: The user interface of a potential controller.

Acknowledgements

Gernot Ruscher's work in the MAike project as well as Frank Krüger's work in the MAxima project are both supported by *Wirtschaftsministerium M-V* at expense of *EFRE* and *ESF*.

6. REFERENCES

[1] http://www.jini.org, OCT 2009.

[2] http://www.knx.org/, OCT 2009.

[3] http://www.ubisense.de, JUN 2009.

[4] http://www.upnp.org/, DEC 2010.

[5] OCZ Technology Group, Inc. Retrieved from http://www.ocztechnology.com, November 2010.

[6] S. Bader, G. Ruscher, and T. Kirste. A middleware for rapid prototyping smart environments. In *Proceedings of the 12th ACM international conference adjunct papers on Ubiquitous computing*, pages 355–356, Copenhagen, Denmark, SEP 2010. ACM.

[7] D. Cook and S. Das. *Smart Environments*. Wiley, 2005.

[8] C. Reisse and T. Kirste. A distributed action selection mechanism for device cooperation in smart environments. In *Proceedings of the 4th International Conference on Intelligent Environments*, Seattle, USA, 2008.

[9] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, 2nd edition edition, 2003.

[10] J. Vidal. Toward direct brain–computer communication. In *Annual Review of Biophysics and Bioengineering*, pages 157–180, 2 1972.