Pers Ubiquit Comput (2006) 10: 136–143
DOI 10.1007/s00779-005-0019-6

ORIGINAL ARTICLE

Olufisayo Omojokun · Jeffrey S. Pierce
Charles L. Isbell · Prasun Dewan

Comparing end-user and intelligent remote control interface generation

Received: 1 March 2005 / Accepted: 29 June 2005 / Published online: 5 November 2005

Abstract Traditional remote controls typically allow users to activate functionality of a single device. Given that users activate a subset of functionality across devices to accomplish a particular task, it is attractive to consider a remote control directly supporting this behavior. We present qualitative and quantitative results from a study of two promising approaches to creating such a remote control: end-user programming and machine learning. In general, results show that each approach possesses advantages and disadvantages, and that neither is optimal.

Keywords Ubiquitous computing · Appliance · User interaction · Machine learning

1 Introduction

Many different kinds of devices, such as TVs, lights, and even vacuum cleaners, offer remote controls for users. These remotes typically contain all the buttons needed for any interaction, the designers believe users might want to have, with the device; however, these traditional remotes fail to address three issues:

1. Interaction spans devices: users often perform tasks involving multiple devices, but remotes are typically built to control single devices. So, users are forced to manage multiple remote controls and switch between them when trying to accomplish a particular task. Consider Fig 1. If this person uses the audio receiver to watch movies in surround sound and also dims the lights in the room via remote control, watching a DVD would require interacting with at least four devices (and possibly more if there are multiple lights in the room to control).
2. Interaction involves a subset of functionality: users typically perform only a small set of tasks with their remotes. These tasks typically do not require all of the functionality that their remote controls provide. Rarely used buttons negatively impact usage by consuming space that could be used to make commonly-used buttons larger and easier to select [3]. Providing too many buttons can also make it more difficult to actually select the desired button [5]. Universal remotes address the issue of interaction spanning devices, but tend to provide coverage of most functionality rather than an appropriate subset.
3. Interaction is idiosyncratic: universal remotes provide a fixed set of buttons designed to provide access to functionality across some predetermined set of devices. In practice, however, users' collections of devices vary widely. Even users with similar device collections may vary widely in how they interact with those devices. The combination of widely varying device collections and interaction patterns makes it difficult for any fixed remote control to provide an effective and enjoyable experience.

These issues can be addressed by a remote with the following characteristics. It allows users to control a heterogeneous collection of devices. It is complete, providing all of the buttons that users need to complete each task. It provides a minimal task-based grouping, presenting all and only the buttons needed to accomplish a single task. Finally, it is efficient, allowing users to activate multiple functions, even across devices, with a single button press.

O. Omojokun (✉) · P. Dewan
University of North Carolina, CB3175 Sitterson Hall,
Chapel Hill, NC, USA
E-mail: omojokun@cs.unc.edu
Tel.: +1-919-9621806
E-mail: dewan@cs.unc.edu
Tel.: +1-919-9621823

J. S. Pierce · C. L. Isbell
Georgia Institute of Technology, 801 Atlantic Avenue,
Atlanta, GA, USA
E-mail: jpierce@cc.gatech.edu
Tel.: +1-404-3854239
E-mail: isbell@cc.gatech.edu
Tel.: +1-404-3854304

Fig. 1 A person's living room containing different devices

By creating such remotes, we can move from remotes with fixed functionality to personalizable remotes. Two potential approaches to realizing this vision are end-user programming [14] and machine learning (ML) [9]. In end-user programming, users manually assign the buttons they believe are sufficient to accomplish their tasks to graphical "screens". They then work with a single, handheld remote control that can display those screens. ML also uses a single, handheld remote to display screens; however, it uses the recorded history of a user's actual remote interactions to infer appropriate groups of buttons for the performed tasks.

In this paper, we compare these two previous approaches together. First, we describe the related work. We then describe in detail the experiments we performed. Finally, we present our conclusions and future work.

2 Related work

Today, there are several commercially available "universal remote controls" that allow end-user programming, such as the Philips Pronto, OmniRemote, and Nevo. The Pronto is typical: the handheld device displays graphical "buttons" that emit an infrared signal associated with a device command. Buttons are grouped on screens, and screens are further grouped by devices. The placement and look of each button, screen and device is completely under user control via programs available for PCs. The Pronto only executes remote control software, but the OmniRemote and Nevo execute on multi-function handheld computers—respectively, the Palm and Compaq Ipaq.

These tools only support infrared devices that exist today. There have also been systems built to support devices of the future—in particular, devices that communicate with one another over networks. These systems allow users to use a mobile device to either: (1) download code that implements device user-interfaces or (2) dynamically generate a user-interface of a given device based on a description of the device's functions. UPnP [11], Jini [18], MOCA [1] are examples of the first approach. CMU's Personal Universal Controller [13], Hodes' System [6], The Universal Remote Console Project [19], and ICrafter [16] are examples of the latter approach.

None of these systems described thus far offer a user-interface that uses ML to adapt to a particular user. Such systems [5, 17, 10, 8] do exist in other domains, and many of them have been surveyed in [9]. The most closely related work to what we describe here is that of [2] and [9]. In the former, an ML algorithm uses a Markov model to predict the next device that a user will want to use. The latter work describes a system that is, to the best of our knowledge, the first example of the ML approach we described in the previous section. Specifically, the system keeps track of the commands executed by a user over time, and each command is then re-represented as a distribution over the commands that follow it, inducing a similarity metric across commands. A straightforward clustering algorithm is then used to discover groups of "similar" buttons. These groups prove to map directly to tasks the user performs. The algorithm is shown to be quite effective in dynamically predicting appropriate individualized button clusters associated with the habits of two distinct users.

3 Experiments

In order to compare the efficacy of the two approaches, we recruited ten participants to participate in a series of experiments. Table 1 summarizes our participants' characteristics.

To gather each participant's interaction history with their remote controls, we logged every remote control command that participants issued for 1 week. While a previous experiment [9] involving two users suggested that a weekend's worth of observations of heavy device usage was enough for a ML algorithm to infer useful button groups, we chose to gather a week's worth of data for each participant because we anticipated that the device usage was likely to vary more widely with more

Table 1 A summary of our ten participants

Participant	Summary			
	Gender	Devices	Age	Education/Employment
P1	Male	TV, VCR, DVD player	25	Final semester masters student in computer science. Full-time computer programmer
P2	Male	TV, DVD changer, Receiver	30	High-school graduate. Food server in restaurant
P3	Female	TV, VCR combo, cable box	23	Second year PhD student in sociology
P4	Male	TV, DVD player	25	Second year medical student
P5	Female	TV, DVD player, stereo	23	College graduate in journalism. Works full-time in advertising
P6	Female	TV, VCR/DVD combo	27	Second year PhD student in biostatistics
P7	Male	TV, VCR, DVD, Receiver, XBOX	27	Masters degree in computer science. Full-time programmer
P8	Female	TV, VCR combo, DVD	26	Second year law student
P9	Male	TV, DVD, stereo	24	College graduate in marketing. Full-time mortgage analyst
P10	Male	TV, cable box w/built-in TIVO	24	College graduate in marketing. Unemployed

participants. We were also interested in observing the performance of ML as the amount of user-data increases, so we logged three users for an additional week.

Before we initiated logging for each user, we specifically asked them to use their devices normally, as if our tool was not in place. We logged commands using a tool (described in [9]) that records their remotes' infrared signals. Examples of devices whose remote controls we recorded include TVs, DVD players, VCRs, stereos, and an XBox game console.

At the end of each logging period, we asked each participant to create paper-based screens consisting of the buttons they felt were sufficient for the tasks they performed. The paper-based survey allowed us to explore the quality of remotes the participants could create without requiring that they learn how to use special purpose software necessary to program actual remotes.

We specifically asked participants to imagine designing a remote containing groups of buttons that they commonly use together. We then showed them an iPAQ PocketPC and told them to imagine that such a device could display the button groups using as many screens as they wished. To create these screens, users simply looked over their actual remote controls for buttons required to complete their tasks, found the equivalent square buttons they wanted from the pool of squares, and stuck the buttons on the appropriate cardboard sheet wherever they wished (Fig. 2). Figure 3 shows an example from one participant (P5).

After participants created initial screens, we told them to imagine new buttons that could invoke sequences of commands that they commonly executed. We asked them to think of such sequences and offered them blank squares so they could create associated buttons. They simply labeled the square (e.g. "TV power + DVD power") and stuck it on the screen where they wished to display it. Such buttons, also called macros, are common in existing end-user programming remotes [12, 14, 15].

Finally, we applied an ML algorithm to each user's interaction history to create button clusters adapted to the particular user. The specific algorithm we used is described above and in much more detail in [9].

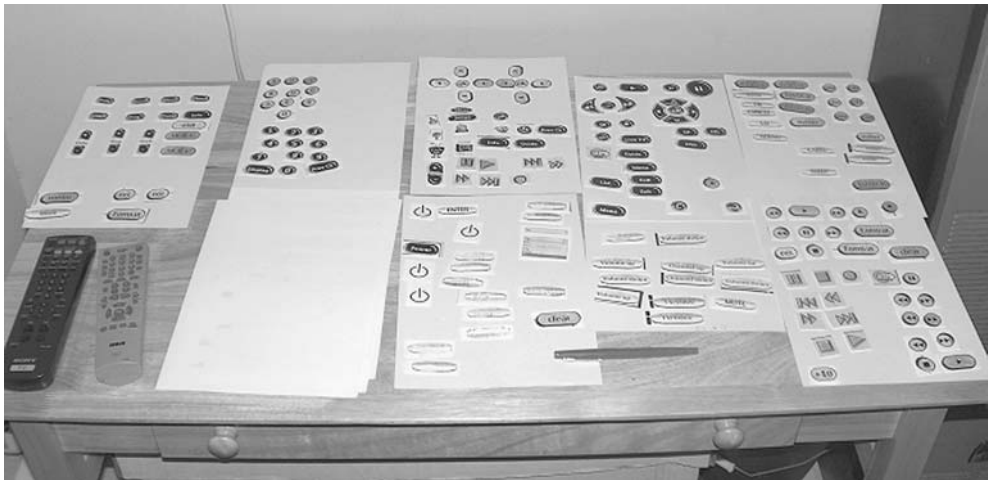

Fig. 2 Our setup containing the participants' remote, several blank sheets, and button squares. Reusable puddy allowed participants to move buttons between sheets. Each sheet represented a single screen

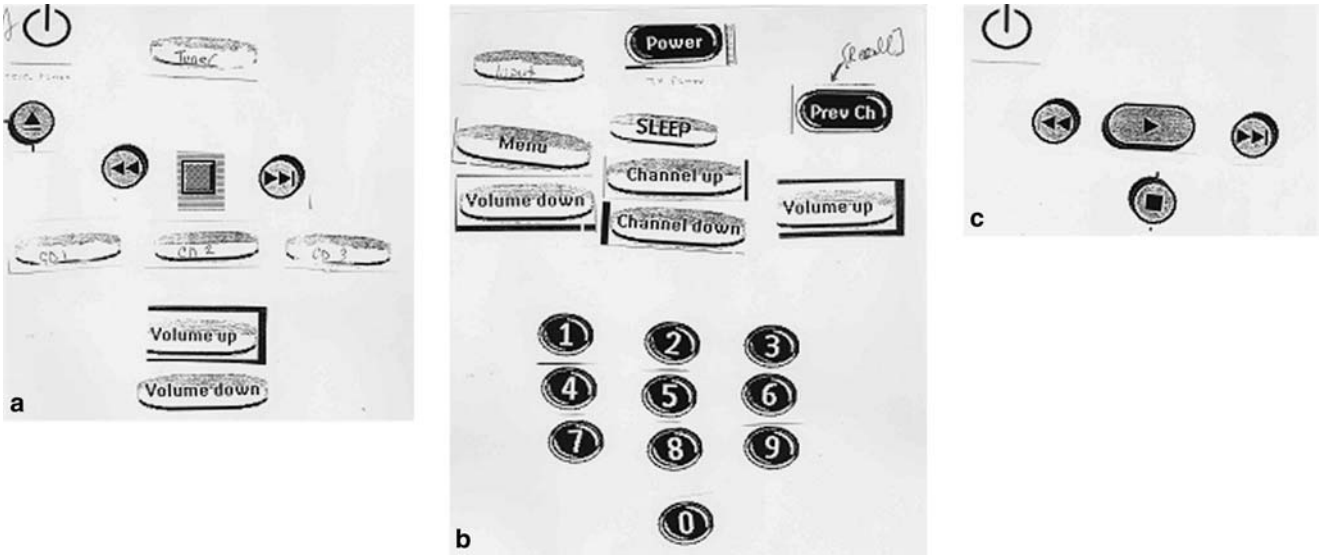

Fig. 3 The three screens for P5

4 Results

We will frame the results of our experiments by comparing how well the two approaches approximated an optimal remote. In particular, we measure:

1. Completeness: an approach should not yield remote controls that are missing required buttons.
2. Task-based grouping: an approach should limit or eliminate switches between remotes or screens when completing a task by properly grouping buttons together.
3. Efficiency: the better the macros an approach can provide, the more efficiently users can complete tasks.

4.1 Completeness

Each approach has different limitations in its ability to yield complete remotes.

4.1.1 End-user programming approach

Seven of the ten participants did not include at least one button that they actually used. The number of missed buttons varied between 0 and 12, with no commonly omitted buttons between users (Table 2).

To illustrate the types of omissions users made, consider the buttons P3 missed. P3 missed the cable box's page-up and page-down buttons for browsing through channel and show listings, despite pushing these buttons 164 and 259 times respectively. In addition, P3 missed the TV channel up and down buttons and the cable box's 'c' button, which the participant uses to set show reminders. Our results suggest that end users are likely to create an incomplete remote control, even if they frequently use all of the buttons that they should include.

4.1.2 ML approach

Because the ML approach draws on an actual record of a user's interactions, it does not omit user commands; however, it has the disadvantage that it only adds buttons that it has seen. Consider that P5 designed a screen containing the CD1, CD2, and C3 buttons of her stereo (Fig. 3)—these buttons play the CD in a given slot (1–3) in the stereo. During logging, she only played the CD in slot 1 by pushing CD1, thus, the ML algorithm did not include the CD2 and CD3 buttons. P5 did mention that all CD slots in the stereo contained a CD throughout the week, but she had only been interested in listening to the CD in slot 1.

4.2 Task-based grouping

We determined the tasks that users most commonly performed, along with the buttons those tasks required, using interviews and analyzing the participants' logs.

Table 2 A count of each participants missed buttons

Participant	Number of missed buttons
P1	12
P2	1
P3	5
P4	2
P5	2
P6	5
P7	0
P8	5
P9	0
P10	0

4.2.1 End-user programming approach

Rather than creating screens associated to specific tasks, half of the participants created screens for single devices. Figure 3 demonstrates this behavior: P5 created individual screens for a stereo (Fig. 3a), TV (Fig. 3b), and DVD player (Fig. 3c) instead of creating a set of screens that spanned those devices. This forces users to switch screens to accomplish common tasks, as shown in Table 3.

Counts include the number of screen switches required to also shut off each device after task completion. For the task of watching a DVD or VHS tape, we also included the action of adjusting the volume because the collected interaction data shows that it was a commonly occurring part of those tasks.

The data shows that only two out of ten participants created a set of screens that do not require switching screens during common tasks. The other users designed screens that require switching screens several times for a given task. We note in particular that participants appear to have been trained to believe that a particular remote control screen should control a single device despite instructions that the views they designed should group buttons that they commonly use together. As a result, the end-user programming approach shares the problem of switching screens or devices with traditional remote controls.

4.2.2 ML approach

In this approach, the ability to find appropriate task groupings depends on the clustering algorithm used. We found that 1 and 2 weeks worth of logging produces a mix of both appropriate and inappropriate clusters. Figure 4 provides an example. It shows a two-dimensional projection of P5's clusters.

On the left side, cluster (a) contains several of the TV channel buttons, which P5 uses together when watching TV. In general, cluster (b) contains the buttons for preparing the VCR to record a future TV show, and cluster (c) contains the buttons for watching VHS tapes on the VCR/DVD combo. These clusters, however, each have buttons that do not belong in the tasks to which they associate. For example, cluster (a) has the VCR/DVD combo's '8' button, which does not belong in a cluster for watching TV. Also, cluster (c) contains the VCR/DVD '7' button, which does not belong in the cluster for watching a VHS tape. Such cases of inappropriately placed buttons can require users to switch between possibly many different task UIs just to complete a single task. In our experiments, there were no users whose data yielded a entire set of clusters in which all buttons within each cluster share a specific (and single) task.

The above example also demonstrates another limitation of the ML algorithm we currently use. It does not support multiple copies of a given button, thus it places the button in only the cluster it believes results in the best fit. To illustrate, the algorithm places the TV power button in cluster (b); however, all tasks require the ability to turn the TV on and off. Augmenting the ML algorithm with heuristics could address this problem, as well as potentially preventing some cases of buttons ending up in inappropriate clusters. One such example could be to always keep a device's (e.g., a TV and VCR) channel number buttons in the same cluster, perhaps the cluster in which most of the numbers exist. Alternatively, we could use a clustering algorithm such as expectation–maximization that does not force hard membership into only one cluster.

In summary, the algorithm's clustering mechanism, which attempts to generate task-based groupings, is not optimal after 1 or 2 weeks of observed interaction. Previous work [9] does show that the algorithm's performance for defining appropriate clusters increases with intensity of use, so presumably this approach would only improve over time; however, it is not likely that users would be willing to wait long enough.

4.3 Efficiency

Both approaches have limitations in their ability to support macros.

Table 3 A count of screen switches required for participants' common tasks

Participant	Task	No. of screen switches	No. of devices required
P1	Watch a VHS tape	4	2
P2	Watch a DVD	3	2
P3	Watch cable TV (includes using the cable box's show listings to find a single interesting show to watch)	5	2
P4	Watch a DVD	4	2
P5	Watch a DVD	4	2
P6	Set up the VCR to record a future TV show	2	2
	Watch a VHS tape	2	2
P7	Watch cable TV (using Tivo)	1	2
	Use XBOX (to listen to MP3 music files)	1	2
P8	Watch a DVD	2	2
P9	Watch a DVD	2	2
P10	Watch cable TV	1	2

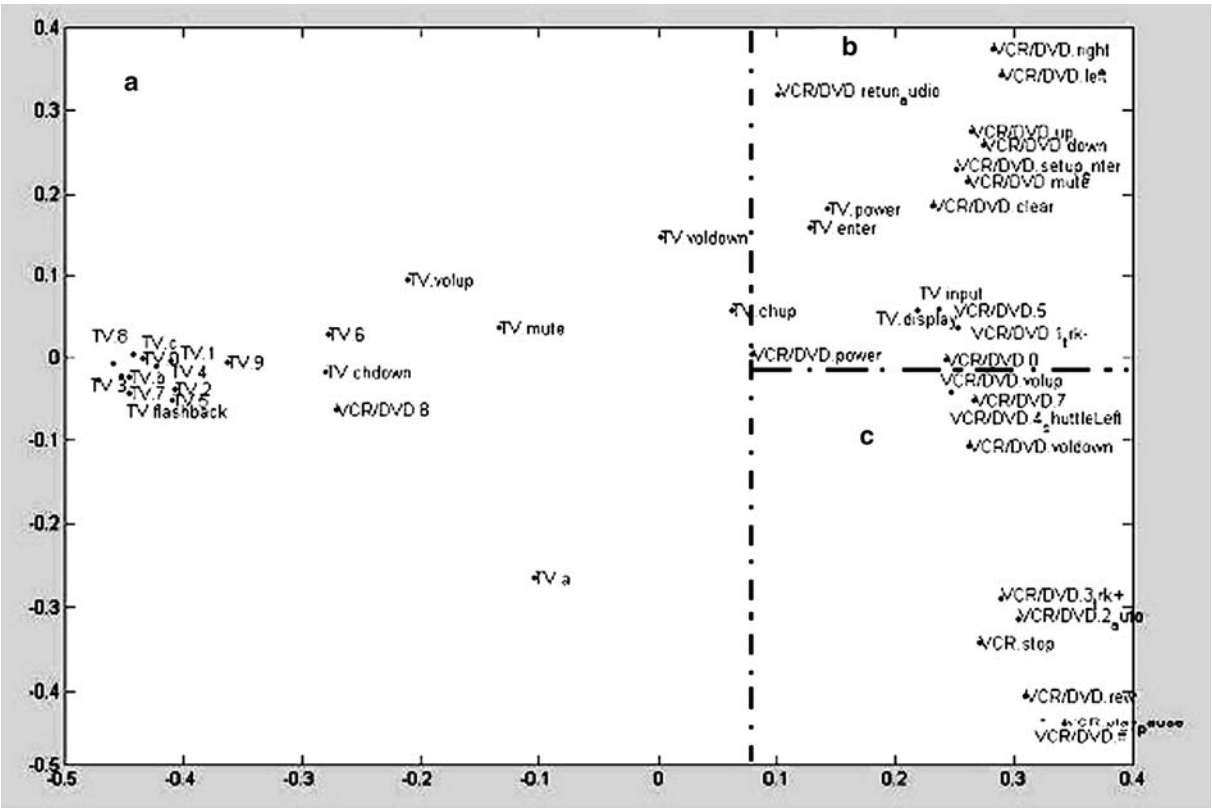

Fig. 4 A projection of P5's clusters

4.3.1 End-user programming approach

All users requested at least one macro. Most of the macros they requested were actually sequences of commands they commonly issued (as the logs show); however, we found that two participants (P2 and P3) requested macros that do not reflect sequences of buttons they actually used. To illustrate, P2 requested two macros for (1) turning on the DVD changer and receiver in order to watch a movie and (2) turning on the DVD changer and receiver and then having the DVD changer play a music CD inside one of its five slots. P2 performed the particular tasks associated with the two macros several times over the week. However, the two sequences did not occur in his entire log. In fact, we found no instance of him pushing the receiver's power button. When asked, the participant admitted that he always left the receiver on. As a result, pushing any of the two macro buttons would actually sidetrack him from performing the desired task because it would result it turning off the receiver.

4.3.2 ML approach

Our ML algorithm can only place buttons in their appropriate task-based clusters.

Because the buttons that make up a macro are inherently part of the same task, they would share the same cluster. The cluster for a given task can thus consist of both macro and non-macro buttons. The algorithm, however, cannot differentiate between the two kinds of buttons and therefore it cannot identify macros on its own.

5 Conclusions and future work

In this paper, we identified several weaknesses of traditional remotes. Based on these weaknesses, we evaluated end-user designed remotes and automatically-generated remotes. The results show that neither is optimal. Users have incorrect models of their own behavior, but automatic methods require lengthy observation to discover accurate models. We believe that a better solution is to combine both the end-user and adaptive approaches to yield a mixed-initiative approach [7] that uses the benefits of one approach to mitigate the limitations of the other, providing better support for:

1. Completeness: because users sometimes miss buttons, our ML algorithm could help provide completeness by validating their designs. Alternately, users could assist the ML algorithm by providing it with a list of buttons whose use it has not observed.
2. Task-based grouping: a possible way of changing the device-centric approach of users is to give them an initial set of task-based clusters provided by an ML

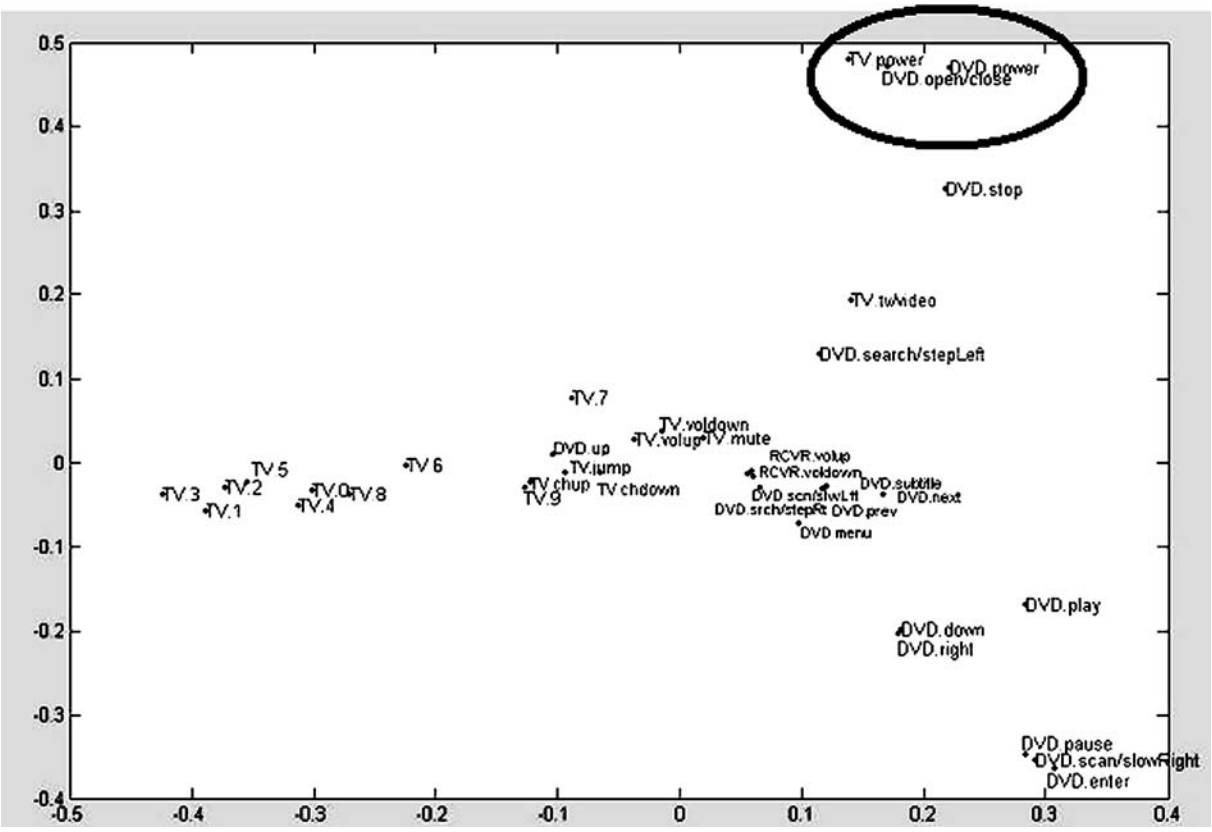

Fig. 5 A projection of P2's clusters

algorithm. Users can then use their intuition to refine these clusters, given that the algorithm sometimes places buttons in wrong clusters.

Users can also design their own remote controls without waiting for the algorithm. Although it is likely that these remote controls will be device based, rather than task-based, they should at least reduce the number of displayed buttons. The ML algorithm could then migrate users toward a more optimal solution by observing users' interactions with their designed remote controls and periodically offering suggestions for new designs when it has observed enough data to be confident in an improvement.

3. Efficiency: a mixed-initiative solution could draw on the clusters from the ML algorithm to validate user-suggested macros, and it could potentially infer possible macros by looking at the button clusters. To illustrate, recall from Sect. 4.3 that the participant requested a macro that would invoke the receiver and DVD power button before watching a movie. The macro implied by the cluster circled in Fig. 5 actually corrects P2's initial intuition by omitting the receiver power button. Furthermore, it adds two commands that P2 did not consider—the TV power button and the DVD open/close button, which would allow the user to place the desired DVD in the device.

The next step in our work is to formally define and develop a complete mixed-initiative system. From there, we can then evaluate its performance characteristics over an extended period across a variety of users.

Acknowledgements This research was funded in part by Microsoft and NSF grants ANI 0229998, EIA 03-03590 and IIS 0312328.

References

1. Beck J, Geffault A, Islam N (1999) MOCA: A service framework for mobile computing devices. In: Proceedings of the international workshop on data engineering for wireless and mobile access
2. Desai N et al. (2002) Automated selection of remote control user interfaces in pervasive smart spaces. In: Pervasive smart spaces HCIC winter workshop
3. Fitts PM (1954) The information capacity of the human motor system in controlling the amplitude of movement. J Exp Psychol 47:381–391
4. Hick WE (1952) On the rate of gain of information. Q J Exp Psychol 4:11–26
5. Hirsh H, Davison B (1997) An adaptive UNIX command-line assistant. In: Proceedings of the first international conference on autonomous agents
6. Hodes T, Katz R (1999) A document-based framework for internet application control. In: Proceedings of the second USENIX symposium on internet technologies and systems (USITS '99)
7. Horvitz E (1999) Principles of mixed-initiative user interfaces. In: Proceedings of CHI '99, ACM SIGCHI conference on human factors in computing systems, Pittsburgh, PA

8. Horvitz E et al (2002) Coordinator: probabilistic forecasting of presence and availability. In: proceedings of the 18th conference on uncertainty and artificial intelligence, pp 224–233
9. Isbell C, Omojokun O, Pierce J (2004) From devices to tasks: automatic task prediction for personalized appliance control. In: proceedings of the 2nd conference on appliance design
10. Isbell C et al (2001) A social reinforcement learning agent. In: proceedings of agents
11. Microsoft corporation. Universal plug and play forum. http://www.upnp.org/
12. Nevo. http://www.mynevo.com/Mynevo/Home.aspx
13. Nichols J et al. (2002) Requirements for automatically generating multi-modal interfaces for complex appliances. In: proceedings of the 4th IEEE international conference on multimodal interfaces
14. OmniRemote. http://www.pacificneotek.com/omnisw.htm
15. Philips Pronto. http://www.pronto.philips.com/
16. Ponnekanti S et al. (2001) ICrafter: A service framework for ubiquitous computing environments. Proceedings of UBICOMP
17. Ruvini J, Dony C (2000) APE: learning user's habits to automate repetitive tasks. In: proceedings of intelligent user interfaces 2000, pp 229–232
18. Sun Microsystems. Jini™ Network Technology. http://www.sun.com/jini/
19. Universal Remote Console project. http://dsonline.computer.org/0403/d/b1sta.htm