Bridging Web Applications and Perceptual User Interfaces

Ingo Lütkebohle, Sven Wachsmuth, and Franz Kummert

Angewandte Informatik, Universität Bielefeld
{iluetkeb,swachsmu,franz}@techfak.uni-bielefeld.de

Abstract. We present a system that bridges the perceptual user interface paradigm and web applications, through automatic analysis of the user interface layout. Perceptual input is mapped to application elements automatically by a new approach, the "Interface Robot", thus dramatically reducing the effort to adapt existing applications. In addition, we demonstrate how knowledge about the user interface provides a powerful, yet easy to obtain, constraint to improve robustness of perceptual interpretation. First evaluation results for the approach are given with respect to a prototypical web application using hand-gestures.

1 Introduction

In many areas, from ubiquitous mobile phones and PDAs, through large displays in public installations, to less common scenarios such as personal robots, the previously dominant interaction devices – keyboard and mouse – have been found impractical or undesirable, due to size and environmental constraints or the amount of attention they require [1]. Perceptual user interfaces (PUIs), in contrast, take advantage of the wide range of human interaction modalities to alleviate some or all of these constraints.

However, input analysis in PUIs poses a much harder problem than the clear and accurate inputs of keyboard and mouse. While recognition of speech and gestures has made great strides in recent years, it has also become clear that their correct interpretation requires (at least) contextual information, as the input itself is often highly ambiguous. Determining context perceptually is often just as hard as the original recognition problem, however, and therefore, a more viable approach is to use application specific knowledge. Traditionally, this has required considerable expert customization for each application [2].

This paper introduces an approach that reduces customization effort by determining the UI context (position, size and function of UI elements) automatically from the active display of web applications. A reusable component dubbed "Interface Robot" will be presented that performs the necessary analysis and control from inside an ordinary web-browser and facilitates communication with the vision system.

While the UI context does not specify what the user *is* doing (in the world) it limits what the user *could* be doing (to the interface). We therefore consider the UI context a valuable constraint for interpretation of perceptual data.

In the remainder of this paper, we will first discuss earlier work in the chosen area, then introduce the web architecture and argue why it is a particularly suitable foundation for perceptual user interfaces. Subsequently, we introduce the method and the resulting architecture for multi-modal interaction. Finally, the evaluation on an example application is described.

1.1 Related Work

Perceptual user interfaces in general are well summarized in [3].

Reducing the effort for PUI development has been addressed more recently by Kjeldsen et. al [4]. They propose a flexible architecture based on XML descriptions of the application UI and also make an important step to "componentize" perceptual inputs by providing a set of generic user interface components. However, the description format is specific to their system and does not make use of existing standards. According to Borkowski et. al [2], the earlier approach also requires considerable expertise in vision systems for adaptation, a drawback which they aim to address in their system through the use of robust vision algorithms. They also provide small, flexible input components that can be combined to create new types of input elements. In both cases, the effort of providing a description of the visual interface is left to the application developer.

In contrast to these systems, the proposed approach re-uses the existing UI specification of web applications, which exists in the form of the standard HTML format. Additionally, initialization of the vision system also makes use of context information, thus adapting to changing conditions automatically.

Using perceptual inputs for existing applications has been advocated by the W3C when, in 2002, a working group was formed to specify multi-modal input formats [5]. This effort addresses only transport, not interpretation, of multi-modal inputs. We provide interpretation in the application context by mapping inputs to the existing interface elements autonomously.

2 Web Technologies for PUIs

This section introduces the most important terms and concepts of the web architecture. The web has seen continuous, rapid development over more than 15 years with several technology generations. Despite this, the basic concepts remain unchanged and we will concentrate on what makes the web architecture suitable for our purposes. Unless otherwise noted, all of the formats and protocols have been specified by the World Wide Web Consortium (W3C) [6].

2.1 Terms and Concepts

The World Wide Web (Web) is a client-server system, where web-browsers (clients) communicate with web-servers via the stateless HTTP protocol. Web documents are given in the Hypertext Markup language (HTML), which embeds machine-readable semantic descriptions in the content, with visual appearance being specified in auxiliary Cascaded Style Sheets (CSS). Examples of semantic descriptions relevant for interaction include “links” and “input” elements. Internally, the browsers represents document elements as well as display attributes (e.g. the position) in the standard Document Object Model (DOM).

2.2 Client-Side Interaction Capabilities

HTML can embed JavaScript, a scripting language, to provide direct interaction with the user and, most importantly for us, inspect and modify the DOM at runtime through the “Interface Robot”. This can affect both content and visual appearance. Additionally, the browser generates events about changes to the interface, such as page load, scrolling, mouse-clicks and so on. Conversely, these can also be synthesized through JavaScript.

While we consider the web architecture very promising, there is one aspect we are concerned about, namely the update speed of the user interface. Web browsers have originally been designed to display static web-pages and interactive content was restricted to video plug-ins or Java applets. At the moment, the so-called “reflow”, that is, the re-layouting and display of the page after a change to the structure, takes considerable time. The exact time depends on the amount of change incured and is thus not precisely predictable. For “small changes” browser developers give an estimate of 100 reflows per second as the maximum attainable [7].

2.3 Reuse and Variability of Interfaces

Adapting existing interfaces to new devices is always a challenge and re-design may be required. Web applications in particular have a history of interface flexibility necessitated due to different web client capabilities. The web architecture has also frequently been extended in its reach, e.g. for mobile devices. Early on, these developments came with specialized standards (e.g. WAP/WML for mobile phones) but over time, both the HTML/CSS combination and the devices themselves have matured to encompass these new applications directly, based on the belief that applications should support varied devices *all at once*. CSS in particular was put forward to separate content and appearance and thus enable versatile display of the same content. Learning from this history, initiatives to increase accessibility [8] and standards compliance [9] have emerged, promoting a disciplined, extensible way of designing interfaces.

We therefore believe that the HTML/CSS combination in particular and the web application space in general are particularly suited as a basis for broadening the reach of perceptual user interfaces. This is not to say that all web applications could be used unchanged, but we predict that quite a few can, and for others, the adaptation effort is most likely considerably reduced.

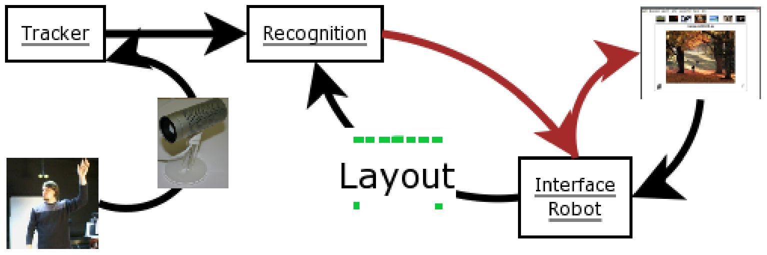


Fig. 1. Components of the Proposed System

3 System Architecture

The defining characteristic of the proposed system is that it augments a pre-existing web application. The following section will demonstrate how a perceptual user interface can be grafted onto an existing application (the “host”-application) without having to modify that application directly.

3.1 Hand Tracking

The most standard part of the proposed system is a hand-tracking component, implemented using the iceWing vision framework [10]. iceWing is a modular image processing architecture based on a Directed Acyclic Graph of processing “plug-ins”. Besides image processing, it also provides network communications functionality, in this case through HTTP.

The image processing steps described in the following provide a simple way for detection of hand motion and show the characteristics necessary to take advantage of the interaction context. We start by detecting motion using temporal differencing, which is a simple technique that subtracts two successive frames, applies a threshold to eliminate noise and binarizes the result to show motion only. Due to its small history of just one frame, it reacts very fast to changes in the scene, such as lighting.

Then, the detected motion is aggregated over a number of interest regions, which are derived from the activatable area of the user interface (see section 4 for details). Additionally, fingertips are detected using a simple fingertip template [11]. Lastly, the motion inside the interest regions is aggregated over the last 10 frames. From those regions that received input over a threshold during this period, and have a fingertip match in them, the highest is selected.

3.2 Interface Robot

The central part of the proposed system is a JavaScript library that we have dubbed “Interface Robot”: It serves a dual purpose:

1. Analyze user-interface to determine size and position of elements.
2. Forward user input from the vision system to the host application.

JavaScript allows us to determine the position of an element on the display in exact pixel coordinates by examining certain attributes of the DOM [12]. This information has to be updated every time the page is scrolled or otherwise changed. To do so, the interface robot registers for page load and scroll events, which are standard in all modern browsers [13]. The gathered information is formatted as an XML message and pushed to the information broker using an HTTP POST request.

In fact, there is much more information available from the DOM tree, not just size and position but also the element-type, formatting and style, the location in the hierarchy and so on. Of these, we currently use the element-type to find activatable elements, specifically, anchors having a HREF-Attribute.

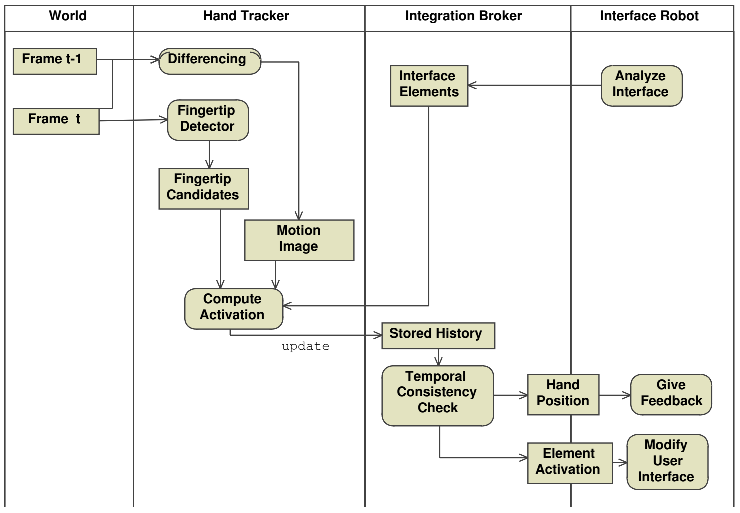


Fig. 2. UML Activity Diagram of the Proposed System

3.3 Information Broker through HTTP

To decouple the Interface Robot from the tracking implementations, we utilize a central message exchange component that distributes messages between all attached components, depending on the content of the message. All messages are formatted as XML and exchanged over HTTP.

Using HTTP allows easy integration with the Interface Robot in the web-browser but unfortunately, the usual request-response style of HTTP client-server interaction does not support delivery of information after the initial response. To achieve high interactivity, we would like the Interface Robot to be notified immediately upon completion of vision processing, without having to resort to polling.

Fortunately, in recent years a new, asynchronous request method informally known as AJaX [14] has gained widespread support in browsers and we have implemented asynchronous notification using AJaX as follows: The interface robot creates an asynchronous HTTP request using the XMLHTTPRequest API and specifies a JavaScript function to be called on receipt of data. In the browser, the request method returns immediately and will not block the UI. Instead, the callback function is called for every chunk of data as it is received, allowing for instantaneous processing.

4 Using Interaction Context During Recognition

In this section, we will describe the constraints that information about the user interface can provide in general and then describe how we make use of it in our hand-gesture prototype.

At first glance, it might appear sufficient to track the hand of the user, acquire its position from frame to frame and use that for interaction, e.g. in place of the mouse position. This simple view unfortunately neglects the initialization, that is, determining *what to track* at any given moment. We call this particular instance of initialization the *selection problem.* Even in the simplest case of one person, one has to deal with two hands already and in our trial experiments, we commonly saw people switch hands during the interaction, e.g. to reach different sides of the large display.

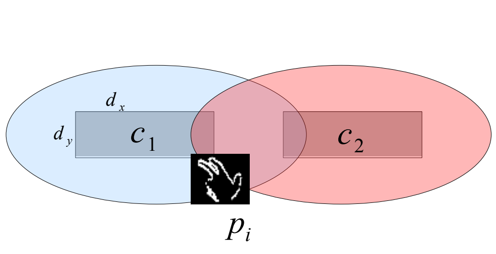
Furthermore, determining hand position only is most likely insufficient to achieve the promise of perceptual user interfaces – after all, if we can only offer a (coarser) mouse-replacement, it will be hard to convince users that this is worth their while. The hand offers many more modalities than just position but as it constantly changes shape, some way of distinguishing meaningful and meaningless motion has to be devised and the proximal context of an interaction element seems a natural distinguishing feature.

4.1 Interaction Context for Gesture Interpretation

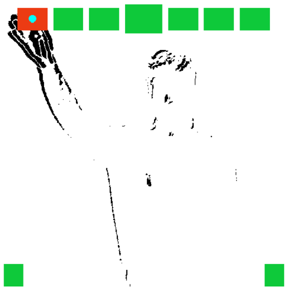
In the proposed prototype, we use the interaction context in two ways: Firstly, we weight fingertips and motion according to proximity to a user interface element, thereby selecting the hand closest to an element. Secondly, we apply a simple motion intensity check to detect "waving"-gestures and use the context to weight the motion by proximity to an interface element. This allows us to get good results from a simple recognizer and at the same time ignore any outside noise.

The effect of the proximity weighting can be seen in figure 3(a), where c_1 and c_2 correspond to interface elements, the surrounding regions to the area considered and p to an example hand match.

The activation is computed as the motion around an element, weighted by element proximity, and proximity of the hand, then choosing the maximum. More formally, let p_i be the x,y-position of the ith hand-candidate and c_j, d_j the center respectively the size of the jth activatable interface element and $m(x, y)$ a binary image that is the output of motion detection. Set the relative position



(a) Interest Regions



(b) Tracking Results

Fig. 3. Context Combination Schema and Display from Application

$r_{ij} := p_i - c_j$ respectively the coordinates $(x, y)_j := (x, y) - c$. The 2d-Gaussian $exp_\sigma(x, y)$ has variance σ. Then the activation a_j is determined by

$$a_j := \max_i \left(\exp_{d_j}(r_{ij}) \cdot \sum_{x_j, y_j} \left[\exp_{d_j}(x_j, y_j) \cdot m(x, y) \right] \right)$$

In the end, the user interface element with the highest activation is chosen.

5 Evaluation

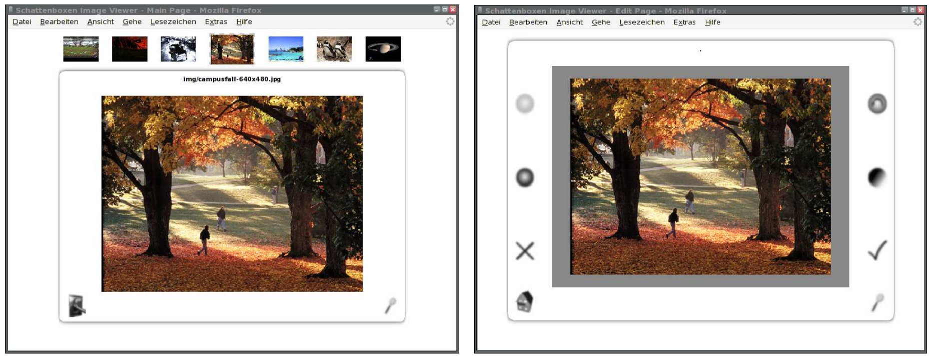
The integration architecture was evaluated on a prototypical application to demonstrate usability and, in particular, gather some insight into the amount and type of feedback required, as we expected that to be the most problematic aspect of using a web-based application.

For the pilot test, the subjects were divided into four groups, using a simple "Thinking Aloud" setup [15], where participants describe their interpretation of the programs behavior during the test. It was complemented by a questionnaire. Test subjects were students from a variety of disciplines, excluding computer science. The test itself took about 5 minutes but we often let subjects "play around" for considerably more time.

In the interest of brevity, the following discussion will be qualitative, quantitative results are available in section 3 of the full study report [16]. A live demo will also be shown during the ICVS 2007.

5.1 Example: Image Viewing and Manipulation

As a testbed for the approach, we applied it to a sample web application that allows the user to browse an image collection and perform basic manipulation operations, such as rotation, contrast adjustment, etc.. See figure 4 for a screenshot.


Schattenboxen Image Viewer - Main Page - Mozilla Firefox
Datei Bearbeiten Ansicht Gehe Lesezeichen Extras Hilfe
img/campusfall-640x480.jpg
Schattenboxen Image Viewer - Edit Page - Mozilla Firefox
Datei Bearbeiten Ansicht Gehe Lesezeichen Extras Hilfe

(a) browse view (b) edit view

Fig. 4. Sample application screenshots from [16]

User Interface As you can see, the application consists out of a large image in the center, a strip of images at the top that serves to change the current image and a number of buttons at the sides of the image that allow further manipulation. The manipulation buttons at the side are spaced widely and therefore easy to distinguish even with coarse position control. The image change area at the top is more of a challenge, because the activatable areas are close together and accidental selection of the wrong button may become an issue.

Feedback We used two types of feedback: Two of the groups received feedback in the form of a large, round cursor, indicating the currently tracked hand position. This cursor was drawn on top of the application by the interface robot.

The two other groups were shown the motion difference image from the camera directly. The image was displayed directly using iceWing, as we had difficulty overlaying it on top of the web interface at sufficient speed. At the moment, we are still investigating better methods for displaying feedback information using web-native technologies.

5.2 Overview of the Experiment

The task given to the participants was to find a certain photographic image using the application and then rotating it by 90 degrees in the edit view.

One group of participants was only informed about the general idea of the application: That it was an image browser operated using gestures. Another group of participants was given specific information about some limitations of the system and its implementation.

5.3 Discussion

Firstly, all groups were able to successfully use the described system and achieve the desired test result. However, it was immediately obvious that both time-to-completion and user satisfaction varied substantially.

Knowledge about Implementation We did not see an influence of knowledge about the implementation. Both the group that received no information except that their hand would be tracked and the group that did receive information about system limitations achieved similar interaction speed.

Type of Feedback There was one issue where the type of feedback made a substantial difference: When users left the area viewable by the camera. In the case of a simple position feedback, this problem was not detected by the user and caused confusion. While the knowledgeable group knew about this potential problem, they still had the same problems in getting back into the viewable area. The group that was shown the motion image, however, managed to avoid this problem entirely by using visual feedback to avoid borders.

Button Activation We initially suggested that people use a "waving"-motion to activate the buttons. This, however, induced a significant simultaneous position change and was difficult to control.

Fatigue People became fatigued very quickly, due to the effort required for holding up a hand above the level of their elbow. We received several requests to position the scroll-bar at the bottom of the application, as this would have been usable while keeping the hand at hip-level.

6 Conclusion

We have shown that not only can web applications be used with perceptual user interfaces (PUIs) but that the user interface analysis they afford provides a powerful constraint for robust pattern analysis. Furthermore, we have introduced a reusable component, the Interface Robot, that allows bridging PUIs and web applications in an automated fashion.

This architecture, we believe, affords two improvements: Firstly, it allows extending the reach of PUIs to a whole new range of applications, easily. Secondly, the use of interface analysis provides a very powerful constraint to improve robustness not just for interpretation of inputs but also for the selection problem and adaptation to novel environments.

Obviously, a number of open issues remain. As the evaluation indicates, the feedback strategy needs to be improved to give users more information about how the system perceives them. We also need to address the fatigue issue, e.g. by allowing to keep the hand at around hip-level. With regard to the input supported, the current prototype analyzes only the application itself. This means that the interaction does not extend to operations afforded by the browser application, such as scrolling and going back in the history. We plan to investigate the use of gestures to synthesize some or all of these events.

A mid-term development will be exploration of adaptive interfaces, that change their display in reaction to the perceptual input. We expect this to ease analysis and improve robustness and user satisfaction. This kind of interface modification on the fly is supported naturally by Web applications through CSS changes and is one of the original motivations that has led us to pursue this direction.

6.1 Acknowledgments

We thank the students in the "Interactive Displays" project for thoughtful discussion and lots of experimental insights: Jörg Eitemüller, Corina Fietz, Jan Hammerschmidt, Andre Homeyer, Daniel Katzberg, Janick Martinez Esturo, Nikita Mattar, Kai Rabien, Frederic Siepmann, Dennies Wiebusch and Christoph Wortmeier. We also gratefully acknowledge the support of the German Ministry of Education and Research – BMBF (project ERBI).

References

1. Oviatt, S.: Ten myths of multimodal interaction. Communications of the ACM **42**(11) (1999) 74–81
2. Borkowski, S., Crowley, J.L., Letessier, J., Berard, F.: User-centric design of a vision system for interactive applications. In: ICVS '06: Proc. of the 4th IEEE Int. Conf. on Computer Vision Systems, IEEE Computer Society (2006)
3. Turk, M., Robertson, G.: Perceptual user interfaces (introduction). Commun. ACM **43**(3) (2000) 32–34
4. Kjeldsen, R., Levas, A., Pinhanez, C.: Dynamically reconfigurable vision-based user interfaces. Machine Vision and Applications **16**(1) (December 2004) 6–12
5. Larson, J.A., Raman, T., Raggett, D.: W3C multimodal interaction activity http://www.w3.org/2002/mmi/.
6. W3C Specifications: http://www.w3.org/.
7. Wilton-Jones, M.: Efficient JavaScript: Repaint and reflow (November 2006) http://dev.opera.com/articles/view/efficient-javascript/.
8. Web Accessibility Initiative: http://www.w3.org/WAI.
9. Web Standards Project: http://www.webstandards.org/.
10. Lömker, F., Wrede, S., Hanheide, M., Fritsch, J.: Building modular vision systems with a graphical plugin environment. In: ICVS '06: Proc. of Int. Conf. on Vision Systems, IEEE (January 2006)
11. Kjeldsen, R., Pinhanez, C and. Pingali, G., Hartman, J., Levas, T., Podlaseck, M.: Interacting with steerable projected displays. In: Proc. of Automatice Face and Gesture Recognition. (2002)
12. Koch, P.P.: Find position http://www.webcitation.org/5MhIsV4he.
13. Koch, P.P.: Event compatibility tables. http://www.webcitation.org/5MhIqat5p.
14. Garrett, J.J.: Ajax: A new approach to web applications (February) http://www.adaptivepath.com/publications/essays/archives/000385.php.
15. Nielsen, J.: Usability Engineering. Academic Press (1993)
16. Lütkebohle, I., Lohse, M.: Experimental results of IDAB experiment on user expectations and feedback performance. Technical report, Applied Computer Science, Bielefeld University (2007)