Context-aware Querying for Multimodal Search Engines

Jonas Etzold[1], Arnaud Brousseau[2], Paul Grimm[1], and Thomas Steiner[2]

[1] Erfurt University of Applied Sciences, Germany,
{jonas.etzold|grimm}@fh-erfurt.de
[2] Google Germany GmbH, ABC-Str. 19, 20354 Hamburg, Germany,
{arnaudb|tomac}@google.com

Abstract. Multimodal interaction provides the user with multiple modes of interacting with a system, such as gestures, speech, text, video, audio, etc. A multimodal system allows for several distinct means for input and output of data. In this paper, we present our work in the context of the I-SEARCH project, which aims at enabling context-aware querying of a multimodal search framework including real-world data such as user location or temperature. We introduce the concepts of *MuSeBag* for multimodal query interfaces, *UIIFace* for multimodal interaction handling, and *CoFind* for collaborative search as the core components behind the I-SEARCH multimodal user interface, which we evaluate via a user study.

Keywords: Multimodality, Context Awareness, User Interfaces

1 Introduction

The I-SEARCH project aims to provide a unified framework for multimodal content indexing, sharing, search and retrieval. This framework will be able to handle specific types of multimedia and multimodal content, namely text, 2D images, hand-drawn sketches, videos, 3D objects and audio files), but also real world information that can be used as part of queries. Query results can include any available relevant content of any of the aforementioned types. This is achieved through Rich Unified Content Annotation (RUCoD), a concept that we have introduced in [14]. It becomes clear that a framework like I-SEARCH faces specific challenges with regards to the user interface (UI). Not only does it have to allow for the combination of multimodal queries, but it also has to do so on different devices, both desktop and mobile. This research being conducted in the context of a European research project, we have time constraints to take into account, hence, we cannot afford to develop two separate UI stacks for desktop and mobile. Instead, we show how using newly added features in the markup language HTML we can kill these two birds with one stone.

The remainder of this paper is structured as follows: Section 2 presents related work, Section 3 introduces our chosen methodology, Section 4 goes into implementation details and presents some preliminary results, Section 5 presents the evaluation of a user study that we have conducted, Section 6 ends the paper with an outlook on future work and provides a conclusion.

2 Related Work

Many have been involved in research to improve user interfaces (UI) for search tasks in the last few years. They widely found evidence for the importance and special demand on the design of search UIs in order to achieve an effective and usable search [6], [16], [9]). Especially with the emerge of the so-called Web 2.0 and the vast amount of user generated content, the raise of the big search engines like Google and Bing continued, and search became one of the main tasks in our daily Internet usage [17]. This trend further increases the importance of the interaction with, and the design of search engines, and also raises the need for extending search tasks beyond textual queries on desktop systems. In this manner, Hearst [7] describes emerging trends for search interface design, which include that interfaces have to be more device-independent (i.e. also support mobile devices), and be able to support the creation of multimodal search queries where text can be enriched with multimedia and real-world data in order to deliver more precise results. With the development of multimodal search interfaces, also concepts for multimodal interaction, as defined by Nigay et al. [13], become an important aspect to distribute all features of this new type of search interfaces to the user. Rigas [18] also found evidence that the use of multimodal features of a search interface, e.g. speech or graphs can support the usability of the whole search engine. In order to combine the efforts towards multimodal interaction, the World Wide Web Consortium (W3C) follows an approach to create a framework that is described by the W3C Multimodal Interaction Working Group with its work-in-progress specification of the "Multimodal Architecture and Interfaces" [2]. Therein, the framework is used to describe the internal structure of a certain interaction component, including the in- and outputs of the various interaction types based on XML. Serrano et al. further created the open interface framework [20], which allows for the flexible creation of combined interaction pipelines using several input channels (e.g. speech and touch). Other approaches to provide frameworks for multimodal interaction and interfaces are described by Sreekanth [21], who uses a *Monitor Agent* to collect events from different modalities and Roscher [19], who uses the Multi-Access Service Platform (MASP), which implements different user interface models for each input modality and is able to combine them to more complex multimodal user interfaces including the synchronization of all inputs along the user interface models.

The possibility to generate more complex, but also more effective search queries with multimodal search interfaces, as well as the nature of the Internet as an environment where people can assist each other, make the integration of collaborative interaction approaches for search engines interesting. Mainly the work of Morris [12] and Pickens [15] described interesting ways of collaborative search approaches. They make use of a search session and state variables in user profiles to transfers changes made in the interface of one user to all other collaborating users and vice versa. Further, the survey about collaborative Web search practices done by Morris [11] as well as the status quo practices presented by Amershi [1] prove the need and practicability of collaborative search methods.

3 Methodology

In this Section, we present our methodology for context-aware querying of multimodal search engines, split up in three sub-tasks: *MuSeBag* for our multimodal query interfaces, *UIIFace* for our multimodal interaction framework, and *CoFind* for our collaborative search framework.

3.1 Multimodal Query Interfaces – MuSeBag

In order to create a visual platform for multimodal querying between user and search engine, the concept of *MuSeBag* was developed. *MuSeBag* stands for **Mu**ltimodal **Se**arch **Bag** and designates the I-SEARCH UI. It comes with specific requirements linked with the need for users to use multiple types of input: audio files or stream, video files, 3D objects, hand drawings, real-world information such as geolocation or time, image files, and of course, plain text. This part of the paper shows the approach chosen to create *MuSeBag*.

Multimodal search engines are still very experimental at the time of writing. When building *MuSeBag*, we tried to look for a common pattern in search-related actions. Indeed, *MuSeBag* remains a search interface at its core. In order for users to interact efficiently with I-SEARCH, we needed a well-known interface paradigm. Across the Web, one pattern is used for almost any and all search related actions: the text field, where a user can focus, enter her query, and trigger subsequent search actions. From big Web search engines such as Google, Yahoo!, or Bing, to intranet search engines, the pattern stays the same. However, I-SEARCH cannot directly benefit from this broadly accepted pattern, as a multimodal search engine must accept a large number of query types at the same time: audio, video, 3D objects, sketches, etc. Some search engines, even if they do not have the need for true multimodal querying, still do have the need to accept input that is not plain text.

As a first example, we consider TinEye[3]. TinEye is a Web-based search engine that allows for query by image content (QBIC) in order to retrieve similar or related images. The interface is split in two distinct parts: one part is a text box to provide a link to a Web-hosted image, while the second part allows for direct file upload (Figure 1). This interface is a good solution for a search engine like TinEye (image input, image output), however, the requirements for I-SEARCH are more complex.

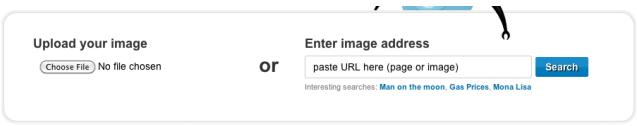

Fig. 1. Screenshot of the TinEye user interface.

[3] `http://www.tineye.com/`

As a second example, we examine MMRetrieval[4] [23]. It brings image and text search together to compose a multimodal query. MMRetrieval is a good showcase for the problem of designing a UI with many user-configurable options. For a user from outside the Information Retrieval field, the UI seems not necessarily clear in all detail, especially when field-specific terms are used (Figure 2).

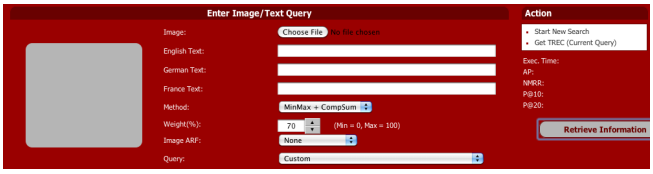

Fig. 2. Screenshot of the MMRetrieval user interface.

Finally, we have a look at Google *Search by image*[5], a feature introduced in 2011 with the same UI requirements as MMRetrieval: combining text and image input. With the *Search by image* interface, Google keeps the text box pattern (Figure 3), while preventing any extra visual noise. The interface is *progressively disclosed* to users via a contextual menu when the camera icon is clicked.

Fig. 3. Input for the *Search by image* user interface.

Even if the *Search by image* solution seems evident, it is still not suitable for I-SEARCH since the interface would require a high number of small icons: camera, 3D, geolocation, audio, video, etc. As a result, we decided to adapt a solution that can be seen in Figure 4.

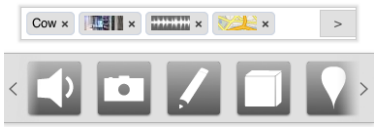

Fig. 4. First version of I-SEARCH interface showing the *MuSeBag* concept.

This interface keeps the idea of a single text box. It is enriched with text auto-completion as well as "tokenization". By the term "tokenization" we refer

[4] http://www.mmretrieval.net
[5] http://googleblog.blogspot.com/2011/06/knocking-down-barriers-to-knowledge.html

to the process of representing an item (picture, sound, etc.) with a token in the text field, as if it was part of the text query. We also keep the idea of *progressive disclosure* for the different actions required by the various modes, e.g. uploading a picture or sketching something. The different icons are grouped together in a separated menu, close to the main search field.

3.2 Multimodal Interaction Handling – UIIFace

Interaction is an important factor when it comes to context-awareness and multimodality. In order to deliver a Graphical User Interface (GUI) that is able to facilitate all the possibilities of a multimodal search engine, a very flexible approach with a rich interaction methodology is needed. Not only the way search queries are build should be multimodal, also the interaction to generate and navigate in such a multimodal interface should be multimodal. To target all those needs, we introduce the concept of *UIIFace* (**U**nified **I**nteraction **I**nter**face**) as general interaction layer for context-aware multimodal querying. *UIIFace* describes a common interface between these interaction modalities and the graphical user interface (GUI) of I-SEARCH by providing a general set of interaction commands for the interface. Each input modality provides the implementation for parts of the commands or all commands defined by *UIIFace*.

The idea of *UIIFace* is based on the open interface framework [20], which describes a framework for the development of multimodal input interface prototypes. It uses components that can represent different input modalities as well as user interfaces and other required software pieces in order to create and control a certain application. In contrast to this approach, *UIIFace* is a Web-based approach implemented on top of modern HTML5 [10] functionalities. Furthermore, it provides a command line interface to the Web-based GUI, which allows for the creation of stand-alone applications outside of the browser window. For the set of uni- and multimodal commands that can be used for I-SEARCH interfaces, the results of Chang [4] as well as the needs derived from the creation of multimodal search queries are used.

Figure 5 depicts the internal structure of *UIIFace* and shows the flow of events. Events are fired by the user's raw input. Gesture Interpreter determines defined gestures (e.g. zoom, rotate) found in the raw input. If no gestures were found, the Basic Interpreter routes Touch and Kinect[6] events to basic cursor and keyboard events. Gestures, speech commands and basic mouse and keyboard events are then synchronized in the Interaction Manager and forwarded as Combined Events to the Command Mapper which maps the incoming events to the defined list of interaction commands that can be registered by any Web-based GUI. The Command Customizer can be used to rewrite the trigger event for commands to user specific gestures or other input sequences (e.g. keyboard shortcuts). This is an additional feature that is not crucial for the functionality of *UIIFace*, but that can be implemented at a later stage in order to add more explicit personalization features.

[6] A motion sensing input device by Microsoft for the Xbox 360 video game console.

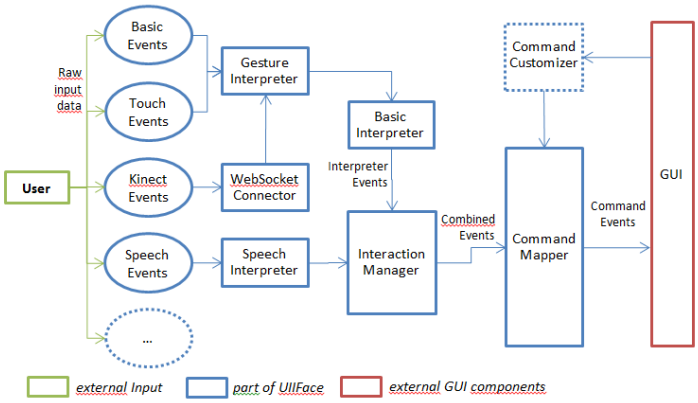

Fig. 5. Schematic view on the internal structure of *UIIFace*.

3.3 Collaborative Search – CoFind

Another part of our methodology targets the increased complexity of search tasks and the necessity to collaborate on those tasks in order to formulate adequate search queries, which lead faster to appropriate result. The increased complexity is primarily caused by the vast amount of unstructured data on the Internet and secondly by situations where the expected results are very fuzzy or hard to describe in textual terms. Therefore the *CoFind* (**Co**llaborative **Find**ing) approach is introduced as a collaborative search system, which enables real-time collaborative search query creation on a pure HTML interface. Real-time collaboration is well-known in the field of document editing (e.g. EtherPad[7], Google Docs[8]); *CoFind* applies the idea of collaborative document editing to collaborative search query composition.

CoFind is based on the concept of shared search sessions in which HTML content of the participants' local clients is transmitted within this session. In order to realize collaborative querying, the concept provides functions for activating collaborative search sessions, joining other online users' search sessions and managing messaging between participants of the search session. Figure 6 shows how the parts listed in the following interact during the search process in order to create a collaborative search session:

Session Manager Controls opening / closing of collaborative search sessions.
Content Manager Broadcast of user interfaces changes to all participants.
Messaging Manager Broadcast of status / user messages to all participants.

The main flow of a collaborative search session can be described as follows: to join a collaborative search session initiated by a user A, a user B must supply the email address of user A. If user A is online and logged in, she receives an on-screen

[7] `https://github.com/ether/pad`
[8] `https://docs.google.com/`

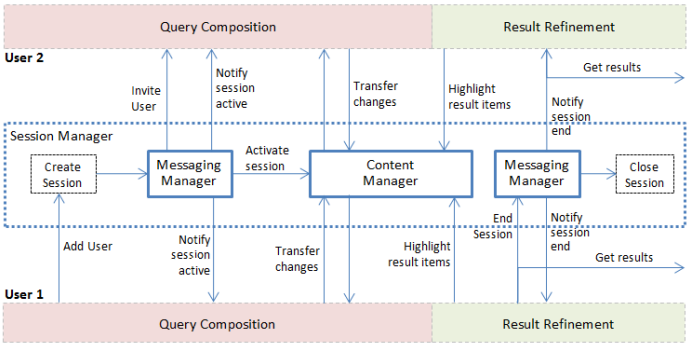

Fig. 6. Schematic diagram of interaction between parts of *CoFind*.

notification and needs to accept the collaboration request of the user B. Upon acceptance, a new session entry is created that stores all participants. Every time a change on the query input field or result set occurs, the changed state is transferred to all participants. Each participant is able to search and navigate through the result set independently from the others, but selected results can be added to collaborative result set. The search session is closed after all users have left the session or have logged out from the system.

4 Implementation Details

The I-SEARCH GUI is built using the Web platform. HTML, CSS, and JavaScript are the three main building blocks for the interface. The rationale behind this choice is that I-SEARCH needs to be cross-browser and cross-device compatible, requirements fulfilled by CSS3 [5], HTML5 [10] and the therein defined new JavaScript APIs that empower the browser in truly novel ways. However, our strategy also includes support for older browsers. When browsing the Web, a significant part of users do not have access to a cutting-edge Web browser. If a feature we use is not available for a certain browser version, two choices are available: either drop support for that feature if it is not important (e.g. drop visual shims like CSS shadows or border-radius), or provide alternate fallback solutions to mimic the experience. We would like to highlight that CSS and HTML are two standards that natively enable *progressive enhancement* thanks to a simple rule: when a Web browser does not understand an HTML attribute, a CSS value or selector, it simply ignores it. This rule is the guarantee that we can build future-proof applications using CSS and HTML. Web browsers render the application according to their capabilities: older browsers render basic markup and styles, while modern browsers render the application in its full glory. Sometimes, however, we have to ensure that all users can access a particular feature. In this case, we use the principle of *graceful degradation*, i.e. use fallback solutions when the technology stack does not support our needs in a certain browser.

4.1 CSS3 Media Queries

The I-SEARCH project needs to be compatible with a large range of devices: desktop browsers, phones, and tablets. Rather than building several versions of I-SEARCH, we use CSS3 media queries[9] to dynamically adapt the layout to different devices.

4.2 Canvas

The `canvas` element in HTML5[10] allows for dynamic, scriptable rendering of 2D shapes and bitmap images. In the case of I-SEARCH, we use `canvas` for user input when the query requires a user sketch, and also to display results in novel ways. The `canvas` element being a core element of I-SEARCH, it is crucial to offer a fallback solution for older browsers. We plan to do so by using FlashCanvas[11], a JavaScript library, which adds the renders shapes and images via the Flash drawing API.

4.3 HTML5 Audio and Video

The `audio` and `video` elements make multimedia content a first class citizen in the Web browser, including scriptability, rotation, rescale, controls, CSS styles, and so forth. For I-SEARCH, this flexibility allows us to create interesting and interactive visualizations of search results. If `audio` and `video` are not available, we fall back to Flash[12] to display media items to users.

4.4 File API

The file API "provides an API for representing file objects in Web applications, as well as programmatically selecting them and accessing their data."[13]. This is interesting in the case of I-SEARCH, since users are very likely to compose their query with local files, like audio files, pictures, etc. The file API allows for a new paradigm to deal with files, such as native support for dragging and dropping elements from the desktop to the I-SEARCH interface. This convenience feature is not crucial, an HTML file upload form serves as a fallback.

4.5 Geolocation

Context-aware search is one of the features of the I-SEARCH framework. This is particularly useful in the case of a user searching on a mobile device, as many

[9] `http://www.w3.org/TR/css3-mediaqueries/`
[10] `http://www.whatwg.org/specs/web-apps/current-work/multipage/the-canvas-element.html`
[11] `http://flashcanvas.net/`
[12] `http://www.adobe.com/products/flashplayer/`
[13] `http://www.w3.org/TR/FileAPI/`

mobile queries are location-based. HTML5 includes the geolocation JavaScript API that, instead of looking up IP address-based location tables, enables Web pages to retrieve a user's location programmatically. In the background, the browser uses the device GPS if available, or computes an approximate location based on cell tower triangulation. The user has to agree for her location to be shared with the application.

4.6 Sensors

Another important aspect for context-awareness is the use of hardware sensors integrated or attached to different device types. These sensors are capable of retrieving the orientation and acceleration of a device or capturing the movements of a user in 3D space. With that knowledge the system is able to make assumptions about the user's direct environment or to detect gestures, which further increases the overall context-awareness. Many of today's mobile devices have accelerometers and gyroscopes integrated that can be accessed through device-specific APIs. HTML5 supports events that target those sensors and defines unified events in the specification for the `deviceorientation` event [3]. Desktop sensors like the Kinect provide depth-information for tracking people in 3D space. These sensors do not yet have a common standard for capturing their data in a browser environment. For those sensors we have created a lightweight WebSocket-based [8] abstraction library.

4.7 Device API

With the Device API [22] the W3C currently creates the next standard related to HTML5. It is mainly targeted to give Web browsers access to attached hardware devices of the client computer. Therefore the Media Capture API, which is a part of the Device API, will enable access to the microphone and the Web camera of the user. We use this API in combination with appropriate fallback routines in order to create audio queries as well as image queries captured on-the-fly.

5 Evaluation

To validate our interface design choices with real multimodal search tasks, we have conducted a user study. We went for a comparative study design to explore how usage of different media types would look like and how they would influence the success rate of search queries. As this user study was mainly focused on the user interface and user interaction parts of I-SEARCH, we assumed that the system always had a correct answer to the (limited) set of permitted queries, even if the real search back-end was not yet in operation at the time of writing. We therefore set the following hypotheses: (H1) Most users will start a search query with just one media type. (H2) Search refinements will be done by adding or removing other media types. (H3) All media types will be handled similarly.

For the user study we recruited seven participants (six male and one female) aged between 20 and 35. All participants were familiar with textual Web-based search. We asked all study participants to find three different items (sound of a tiger, 3D model of a flower, image of a car). The participants were shown these items beforehand in their original format and were then instructed to query the system in order to retrieve them via I-SEARCH. For the study a Windows laptop with a capacitive touch display was used. Each participant was interviewed after the completion of the study. Our goal was to validate our interface design as well as to measure the impact of the possibility of multimodal search. In general, we observed that the concept of multimodal search was new and unfamiliar to all participants. Actually, before the user study all participants considered Web search equal to text-based search, and only by using I-SEARCH they became aware of the possibility to use different media types and of multimodal interaction at all. Our hypothesis (H1) was statistically not supported. It depends highly on the behavior of each individual person whether one or more search items or media types are used. In combination with (H2), one obvious conclusion of the participant interviews was that adding search items as well as customizing them has to be as easy as possible. The participants did not hit obstacles in using one special query modality, however stated that if a query modality was difficult to use, they would replace it by using different query modalities, even if this implied that the search query would become complicated and challenging. The same conclusion applies to hypothesis (H3). In order to allow for multimodal search queries, the following recommendations can be derived from our user study:

1. No query modality should be privileged.
2. The handling of all search modalities should be as consistent as possible.
3. Search refinement should be possible in the result presentation.

6 Conclusion and Future Work

(Tom)

7 Acknowledgment

This work is partly funded by the EU FP7 I-SEARCH project under project reference 248296.

References

1. S. Amershi and M. R. Morris. Co-located Collaborative Web Search: Understanding Status Quo Practices. In *Proceedings of the 27th International Conference on Human Factors in Computing Systems*, CHI EA '09, pages 3637–3642, New York, NY, USA, 2009. ACM.

2. J. Barnett and M. I. W. Group. Multimodal Architecture and Interfaces, July 2011. http://www.w3.org/TR/mmi-arch.
3. S. Block and A. Popescu. DeviceOrientation Event Specification, Editor's Draft, 2011. http://dev.w3.org/geo/api/spec-source-orientation.html.
4. J. Chang and M.-L. Bourguet. Usability Framework for the Design and Evaluation of Multimodal Interaction. In *Proceedings of the 22nd British HCI Group Annual Conference on People and Computers: Culture, Creativity, Interaction – Volume 2*, BCS-HCI '08, pages 123–126. British Computer Society, 2008.
5. Elika J. Etemad. Cascading Style Sheets (CSS) Snapshot 2010, W3C Working Group Note 12 May 2011. http://www.w3.org/TR/CSS/.
6. M. A. Hearst. *Search User Interfaces*. Cambridge University Press, New York, NY, USA, 1st edition, 2009.
7. M. A. Hearst. Emerging Trends in Search User Interfaces. In *Proceedings of the 22nd ACM conference on Hypertext and Hypermedia*, HT '11, pages 5–6, New York, NY, USA, 2011. ACM.
8. I. Hickson. W3C – The WebSocket API Editor's Draft, 2011. http://dev.w3.org/html5/websockets.
9. S.-T. Huang, T.-H. Tsai, and H.-T. Chang. The UI issues for the search engine. *11th IEEE International Conference on ComputerAided Design and Computer Graphics*, pages 330–335, 2009.
10. Ian Hickson. HTML5, A Vocabulary and Associated APIs for HTML and XHTML, W3C Working Draft 25 May 2011. http://www.w3.org/TR/html5/.
11. M. R. Morris. A Survey of Collaborative Web Search Practices. In *Proceedings of the 26th Annual SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 1657–1660, New York, NY, USA, 2008. ACM.
12. M. R. Morris and E. Horvitz. SearchTogether: an interface for collaborative web search. In *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology*, UIST '07, pages 3–12, New York, NY, USA, 2007. ACM.
13. L. Nigay. Design Space for Multimodal Interaction. In *IFIP Congress Topical Sessions*, pages 403–408, 2004.
14. P. Daras, A. Axenopoulos, V. Darlagiannis, D. Tzovaras, X. Le Bourdon, L. Joyeux, A. Verroust-Blondet, V. Croce, T. Steiner, A. Massari, and others. Introducing a Unified Framework for Content Object Description. International Journal of Multimedia Intelligence and Security, Special Issue on "Challenges in Scalable Context Aware Multimedia Computing", accepted for publication, 2010. http://www.inderscience.com/browse/index.php?journalID=359.
15. J. Pickens, G. Golovchinsky, C. Shah, P. Qvarfordt, and M. Back. Algorithmic Mediation for Collaborative Exploratory Search. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, pages 315–322, New York, NY, USA, 2008. ACM.
16. W. Quesenbery, C. Jarrett, I. Roddis, S. Allen, and V. Stirling. Designing for Search: Making Information Easy to Find. Technical report, Whitney Interactive Design, June 2008.
17. W. Quesenbery, C. Jarrett, I. Roddis, V. Stirling, and S. Allen. The Many Faces of User Experience (Presentation), June 16-20, 2008, Baltimore, Maryland, USA, 2008. http://www.usabilityprofessionals.org/.
18. D. Rigas and A. Ciuffreda. An Empirical Investigation of Multimodal Interfaces for Browsing Internet Search Results. In *Proceedings of the 7th International Conference on Applied Informatics and Communications*, pages 194–199, Stevens Point, Wisconsin, USA, 2007. World Scientific and Engineering Academy and Society (WSEAS).

19. D. Roscher, M. Blumendorf, and S. Albayrak. A Meta User Interface to Control Multimodal Interaction in Smart Environments. In *Proceedings of the 14th International Conference on Intelligent User Interfaces*, IUI '09, pages 481–482, New York, NY, USA, 2009. ACM.
20. M. Serrano, L. Nigay, J.-Y. L. Lawson, A. Ramsay, R. Murray-Smith, and S. Denef. The Open Interface Framework: A Tool for Multimodal Interaction. In *CHI '08 Extended Abstracts on Human Factors in Computing Systems*, pages 3501–3506, New York, NY, USA, 2008. ACM.
21. N. S. Sreekanth, S. N. Pal, M. Thomas, A. Haassan, and N. K. Narayanan. Multimodal Interface: Fusion of Various Modalities. *International Journal of Information Studies*, 1(2), 2009.
22. W3C. W3C – Device APIs and Policy Working Group, 2011. `http://www.w3.org/2009/dap`.
23. K. Zagoris, A. Arampatzis, and S. A. Chatzichristofis. www.MMRetrieval.net: A Multimodal Search Engine. In *Proceedings of the 3rd International Conference on SImilarity Search and Applications*, SISAP '10, pages 117–118, New York, NY, USA, 2010. ACM.