Fusion in Multimodal Interactive Systems: An HMM-Based Algorithm for User-Induced Adaptation

Bruno Dumas
WISE Lab
Vrije Universiteit Brussel
Pleinlaan 2
1050 Brussels, Belgium
bdumas@vub.ac.be

Beat Signer
WISE Lab
Vrije Universiteit Brussel
Pleinlaan 2
1050 Brussels, Belgium
bsigner@vub.ac.be

Denis Lalanne
DIVA Group
Université de Fribourg
Boulevard de Pérolles 90
1700 Fribourg, Switzerland
denis.lalanne@unifr.ch

ABSTRACT

Multimodal interfaces have shown to be ideal candidates for interactive systems that adapt to a user either automatically or based on user-defined rules. However, user-based adaptation demands for the corresponding advanced software architectures and algorithms. We present a novel multimodal fusion algorithm for the development of adaptive interactive systems which is based on hidden Markov models (HMM). In order to select relevant modalities at the semantic level, the algorithm is linked to temporal relationship properties. The presented algorithm has been evaluated in three use cases from which we were able to identify the main challenges involved in developing adaptive multimodal interfaces.

Author Keywords

Multimodal interaction, multimodal fusion, HMM-based fusion, user interface adaptation

ACM Classification Keywords

H.5.2 Information Interfaces and Presentation: User Interfaces—*Graphical user interfaces, Prototyping, Theory and methods*

General Terms

Algorithms; Human Factors.

INTRODUCTION

Multimodal interaction has been shown to enhance human-computer interaction by providing users with an interaction model that is closer to human-human interaction than standard interaction via mouse or keyboard. By providing users with a number of different modalities and by offering them the possibility to use these modalities either in a complementary or in a redundant manner, multimodal interfaces take advantage of the parallel processing capabilities of the human brain [25]. Furthermore, due to the different ways on how modalities can be combined, multimodal interfaces have the potential to adapt to a user, by offering them one or multiple preferred schemes of interactions. However, research on the combination or fusion of different modalities has progressed slower than the appearance of new modalities. In particular, fusion-related topics such as the dynamic adaptation of fusion engines, for example with help of machine learning techniques, still need to be addressed in more detail by the scientific community [21].

In this article, we present a novel algorithm for the fusion of input modalities in multimodal interfaces. In the past few years, we focussed on studying different aspects of multimodal input fusion. To this end, a framework for the prototyping of multimodal interfaces, called HephaisTK, has been developed. HephaisTK allows developers to focus on the creation of their multimodal applications, without having to worry about the integration of input libraries, input data normalisation, concurrency management or the implementation of fusion algorithms. The HephaisTK framework comes along with a number of fusion algorithms for managing input data. Among the different algorithms, a symbolic-statistical fusion algorithm based on hidden Markov models (HMMs), which offers the possibility to take a user's feedback into account, has been developed. While this novel multimodal fusion algorithm represents a first major contribution of our paper, a second contribution is the identification of a number of challenges to be overcome for the introduction of user-induced automatic multimodal interface adaptation based on some of our initial explorations.

We start by presenting related work on prototyping tools and multimodal fusion algorithms with respect to the adaptation to the user and context. The following section introduces the HephaisTK framework and its architecture, as well as the SMUIML language, on which the presented algorithm has been built. The HMM-based fusion algorithm that we used for the user adaptation is then presented and we describe how to integrate such an algorithm in a typical multimodal system. This is followed by an evaluation of our HMM-based algorithm. Finally, we provide some conclusions and outline possible future work.

RELATED WORK

Research on fusion algorithms and prototyping tools for multimodal interactive systems has been an active field during the last two decades. In this section, we are going to introduce the state of the art on fusion algorithms for multimodal interactive systems, provide a summary of notable results on prototyping tools and give an overview on adaptation to user and context.

Fusion of Multimodal Input Data

On a conceptual level, Sharma et al. [29] consider the data, feature and decision levels for the fusion of incoming data. Each fusion scheme can operate at a different level of analysis of the same modality channel. *Data-level fusion* considers data from a modality channel in its rawest form, where fusion is generally used on different channels of the same modality, in order to enhance or extract new results. Adaptation of data-level fusion typically focusses on taking into account contextual information to improve results. *Feature-level fusion* is used on tightly coupled, synchronised modalities, such as speech and lip movement. Low-level features extracted from raw data are usually processed with machine-learning algorithms. Adaptation on the feature level can be used when multiple data sources provide data for the same modality. *Decision-level fusion* is the most common type of fusion in interactive multimodal applications, because of its ability to extract meaning from loosely coupled modalities. Complex forms of adaptation can be applied on the decision-level, including the adaptation to the number of users, user profiles or device. On an algorithmic level, typical decision-level fusion algorithms are frame-based fusion, unification-based fusion and hybrid symbolic-statistical fusion.

- *Frame-based fusion* [18] uses data structures called frames or features for the meaning representation of data coming from various sources or modalities.
- *Unification-based fusion* [30] is based on the recursive merging of attribute/value structures to obtain a logical meaning representation. However, both frame-based as well as unification-based fusion rely on a predefined and non-evolutive behaviour.
- *Symbolic-statistical fusion* [6, 31] is an evolution of standard symbolic unification-based approaches, which adds statistical processing techniques to the fusion techniques described above. These hybrid fusion techniques have been demonstrated to achieve robust and reliable results. A classical example of a symbolic-statistical hybrid fusion technique is the Member-Team-Committee architecture used in Quickset [9]. However, these techniques need training data specific to the targeted application.

On a modelling level, the CARE properties as defined by Coutaz and Nigay [10] show how modalities can be composed. Note that CARE stands for *complementarity*, *assignment*, *redundancy* and *equivalence*. Complementary modalities will need all modalities for the meaning to be extracted, assigned modalities allocate one and only one modality to each meaning, redundant modalities state that all modalities can lead to the same meaning and equivalent modalities assert that any modality can lead to the same meaning. The difference between redundancy and equivalence is the way in which cases with multiple modalities occurring at the same time are dealt with.

Multimodal Authoring Frameworks

Quickset by Cohen et al. [9] is a speech/pen multimodal interface based on the Open Agent Architecture[1], which served as a test bed for unification-based and hybrid fusion methods. The integration of multimodal input fusion with the modelling of multimodal human-machine dialogues was introduced by IMBuilder and MEngine [4], which both make use of finite state machines. In their multimodal system, Flippo et al. [14] use a parallel application-independent fusion technique, based on a software agent architecture. In their system, fusion has been realised by using frames. After these original explorations, the last few years have seen the appearance of a number of fully integrated tools for the creation of multimodal interfaces. Among these tools are comprehensive open source frameworks such as OpenInterface [28] and Squidy [20]. These frameworks share a similar conceptual architecture with different goals. While OpenInterface targets pure or combined modalities, Squidy was created as a particularly effective tool for streams composed of low level data. Finally, in contrast to the linear, stream-based architecture adopted by most other solutions, the Mudra [16] framework adopts a service-based architecture. However, few of these tools provided services to test, use and compare multiple fusion algorithms.

Adaptation to User and Context

When considering user adaptation, different ways of adapting the user interface can be considered. On the one hand, the user interface can be created in such a way that it will adapt automatically and without any user intervention. The adaptation can thus be performed automatically to the context of use or to the user. On the other hand, users can be given the possibility to modify the user interface in a proactive way. More formally, Malinowski et al. [23] proposed a complete taxonomy for user interface adaptation. Their taxonomy is based on four different stages of adaptation in a given interface, namely *initiative*, *proposal*, *decision* and *execution*. Each of these adaptations at the four different stages can be performed by the user, the machine or both. López-Jaquero et al. proposed the ISATINE framework [22]. This framework introduces seven stages of adaptation including the goals, the initiative, the specification, the application, the transition, the interpretation as well as the evaluation of adaptation. Interestingly, while adaptation has been investigated for traditional WIMP interfaces [5] or context-aware mobile interfaces [1], less work has been devoted to user-induced adaptation in the context of multimodal interfaces. Octavia et al. [24] explored adaptation in virtual environments, especially on how to build the user model.

The HephaisTK framework that we are going to present in the next section and in particular a novel adaptive multimodal fusion algorithm, contribute to this currently not much explored research on user-induced adaptive multimodal interfaces.

HEPHAISTK FRAMEWORK

Before we present our new solution for user-induced adaptation in multimodal interaction, we introduce the framework in

[1]http://www.openagent.com

which the new algorithm has been integrated, since its architecture, modules and scripting language influenced the choice we made in terms of our user adaptation solution.

HephaisTK is a framework that has been built to help developers in creating multimodal interfaces via a range of tools [12]. The HephaisTK framework shown in Figure 1 offers engineers a framework which manages, stores and presents data coming from a number of input recognisers in an uniform way, with the possibility to query past input events. Second, the tool provides different multimodal input data fusion algorithms, including a classical implementation of frame-based fusion as well as our novel algorithm presented in the next section. Third, multimodal human-machine dialogue is described by means of a high level modelling language, called SMUIML, which is linked to the CARE properties [10]. Compared to the tools in the related work section, HephaisTK focusses on the study of fusion algorithms, as well as on the high level modelling of multimodal human-machine interaction through the SMUIML language.

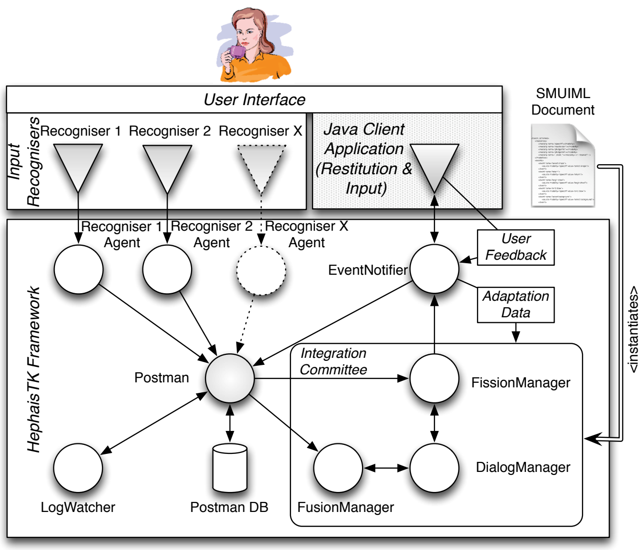


Figure 1. HephaisTK architecture

Developers of multimodal interfaces define the behaviour of their multimodal interface in a configuration file loaded by HephaisTK. This configuration file defines which recognisers to use (*input/output level*), how an event is triggered and how it is linked to the human-machine dialogue (*event level*), and the various states of the dialogue and the transitions between these states are specified (*dialogue level*). The *adaptation level* links the dialogue with contextual and user information. To model this human-machine dialogue for a specific client application, the SMUIML language has been developed [13].

The SMUIML language is divided in three different layers dealing with different levels of abstraction as shown in Figure 2. The lowest level specifies the different modalities which will be used in the context of an application, as well as the particular recognisers to be used to access the different modalities. The middle level addresses input and output events. Input events are called *triggers* whereas output events are called *actions*. Triggers are defined per modality and therefore not directly tied to specific recognisers. They can express different ways to trigger a particular event. For example, a speech trigger can be defined in such a way that "clear", "erase" and "delete" will all lead to the same event. Actions are the messages that the framework will send to the client application. The highest level of abstraction in Figure 2 describes the actual human-machine dialogue by means of defining the contexts of use and interweaving the different input events and output messages between those different contexts, as well as link to adaptation-related information. The resulting description takes the form of a state machine, in a similar way as Bourguet's IMBuilder [4]. The combination of modalities is defined based on the CARE properties as well as the (non-)sequentiality of input triggers.

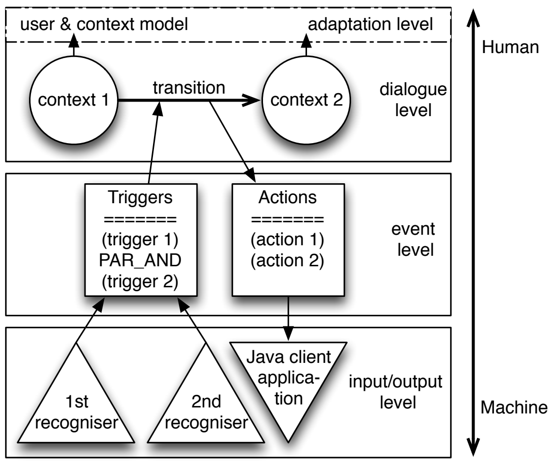


Figure 2. SMUIML and its three levels of abstraction

To come back to the overall HephaisTK architecture, an integration committee is in charge of interpreting the multimodal input data in order to create a suitable answer for the user. The different components of the integration committee are shown in Figure 1. The `FusionManager` and `FissionManager` are linked to the `DialogManager` which informs them about the current state of the interaction and what they can expect as input information. The `DialogManager` relies on the interpretation of the application-specific SMUIML script for managing the interaction. The `FusionManager` focusses on the interpretation of incoming input data based on information provided by the `DialogManager`. It also encapsulates the different available fusion algorithms. In addition to a classic frame-based multimodal fusion algorithm, we have developed an HMM-based multimodal fusion algorithm which is described in the next section.

HMM-BASED MULTIMODAL FUSION ALGORITHM

As presented in the previous section, the HephaisTK framework has been designed to serve as a test platform for multimodal input data fusion algorithms. Based on HephaisTK, different algorithms including our new machine learning-based fusion algorithm have been implemented and tested.

The idea of mixing the adaptiveness of machine learning-based recognition algorithms with the expressive power of higher level rules is getting more attention recently, with examples demonstrating the overall effectiveness in recognition rates and expressiveness of such hybrid approaches [7]. However, symbolic-statistical approaches, such as the ones mentioned in the related work section, tend to not take into account temporal relationships between modalities, which we believe is a necessary feature of fusion algorithms for multimodal interactive systems. Furthermore, the adaptation to user behaviour has only been sparsely studied. We also have to stress that machine learning algorithms have mainly been used for multimodal input fusion at the feature level to typically improve the recognition results of a specific modality or in offline analysis systems such as biometrics or meeting browsers. However, at the decision level, purely rule-based approaches such as frame-based or unification-based fusion have been the norm so far.

Goals of the Algorithm

When considering which type of machine learning should be used, hidden Markov models have been prioritised because of their easy adaptation to time-related processes. Hidden Markov models have historically been used in temporal pattern recognition tasks, such as speech [19, 26], handwriting or gesture recognition. Since the fusion of input data also focusses on time-dependant patterns, HMMs have been seen as one of the obvious choices when considering the different alternatives among statistical models. Due to the fact that SMUIML models the human-machine interaction via states, the transition from the dialogue model to a Markov process is relatively straightforward. Hidden Markov models have already been used for tasks such as video segmentation [17] or biometric person recognition [8] using multimodal input data at the data or feature level. However, to the best of our knowledge, HMMs have not yet been applied for the fusion of multimodal input events at the decision level in the context of real-time multimodal human-machine interfaces.

Our goals when designing the new multimodal fusion algorithm were threefold:

1. Consider a class of machine learning algorithm which has shown its efficiency at modelling the flow of time events and thus is ideally suited for the modelling of multimodal human-computer interaction.
2. In the fusion recognition process take into account a set of the most likely results from probability-based modality recognisers, such as speech or gesture recognisers.
3. Be able to adapt and correct results from the fusion algorithm on-the-fly based on user feedback.

Hidden Markov Models

As presented by Rabiner [26], hidden Markov models are based on discrete Markov processes, in particular discrete time-varying random phenomenons for which the Markov property holds. In place of directly observing the states of the discrete Markov process, a hidden Markov model observes the (hidden) states through a set of stochastic processes that produce the sequence of observations. The most likely sequence of states, given a set of observations, is extracted by the Viterbi algorithm [15]. Finally, the training of hidden Markov models is achieved with help of the Baum-Welch algorithm [2], a particular case of a generalised expectation-maximisation algorithm.

Algorithm in Action

The first challenge when integrating HMMs as fusion algorithms for multimodal interaction is to map the features and states to the actual human-machine interaction model. In our case, data fed to the HMM will be semantic-level information, such as a speech utterance or the high-level description of a gesture such as "flick left". For our implementation of the algorithm, the Java Jahmm HMM library[2] was used. Scripts written in the SMUIML modelling language [13] serve as high-level descriptions of the HMMs topology. A second challenge was to minimise the need for training the algorithm before being able to actually use the system. As we explain later, a pre-training of our system is achieved through the simulation of expected inputs described in SMUIML.

As an example to explain how the algorithm works, let us consider the *"put that there"* example by Bolt [3]. In this example, a user is seated in front of a wall on which different shapes are displayed. To move a shape from one place to the other, the user points to the shape and utters *"put that there"* while pointing to another place on the wall. Five different input events are expected, including three speech related triggers (*"put"*, *"that"* and *"there"*) and two gesture related triggers (i.e. pointing events). Note that in our example, the two pointing events will be considered as events of the same type.

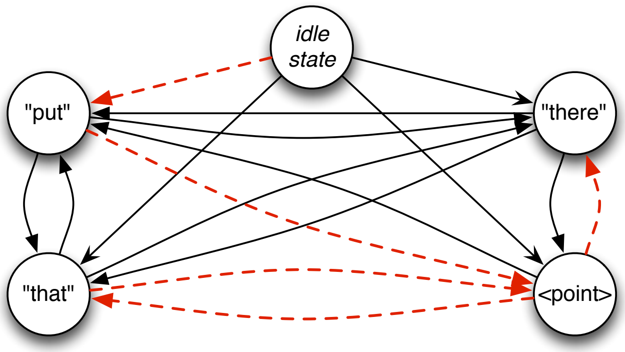


Figure 3. A sequence of states for a *"put that there"* multimodal event

The step from the high-level human-machine interaction description to the actual implementation of the HMM-based fusion algorithm is achieved as described in the following. In SMUIML, the `<context>` element is used to describe different application states. For example, in a drawing application, free line drawing and shape editing could be modelled as different states of the application. Different application states in SMUIML are modelled with help of one HMM for each state. This implies that changing states in an application created with HephaisTK, corresponds to changing from one

[2]http://code.google.com/p/jahmm/

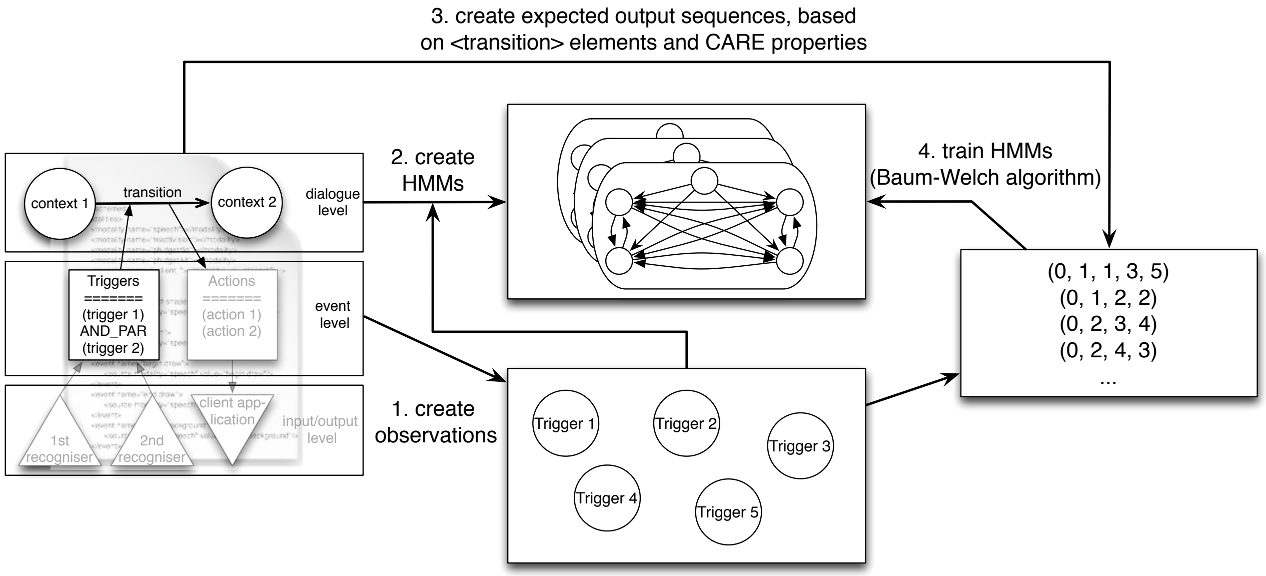


Figure 4. Instantiation of the HMM-based fusion algorithm with observations and HMMs created and trained based on a SMUIML script

`<context>` element to another in SMUIML. This in turn is equivalent to switching from one HMM to another HMM. The *"put that there"* example consists of a single interaction with only one context and thus a single HMM. In the current implementation of the algorithm, any adapted training done after the initialisation phase will not transfer directly during context switching.

At runtime, input events such as a speech utterance or a gesture are modelled as individual observations. Basically, one input event described in the SMUIML script corresponds to one observation of the HMM. Every time an input event is processed by the HephaisTK framework, it is sent to the `FusionManager`. For the list of all input events happening in a specified time window, the `FusionManager` considers potential combinations. Those combinations are then injected into the HMM. The Viterbi algorithm is subsequently used on the HMM to extract the most probable state sequences. These state sequences are compared to a set of expected observation sequences which are defined by the `<transition>` elements of the current `<context>` in the SMUIML script. For example, the following succession of input events: *"put"* *point* *"that"* *point* *"there"* would generate a state sequence as illustrated by the red dashed arrows in Figure 3. If a match is found, the information that a sequence of events corresponding to the SMUIML script description has been found is passed to the client application.

Instantiation of the HMM-Based Fusion Algorithm

The instantiation of HMM-based fusion in HephaisTK is performed based on a given SMUIML script as highlighted in Figure 4. First, all triggers defined in the script are parsed and states are created based on this list of triggers (step 1). Basically, one state corresponds to one trigger definition, with the addition of an `idle` state which serves as a starting state. Then, for each defined `<context>` element, one HMM is instantiated with the number of states corresponding to $T + 1$; T being the number of triggers declared for the current `<context>` (step 2). All states are initially interconnected, except the `idle` one. A discrete output observation distribution is used to connect the hidden layer with the observations. All `<transition>` elements declared in the SMUIML script are then parsed and, based on the synchronicity rules, a set of all expected output sequences is generated for each context (step 3).

The input events processed by HephaisTK all come from users interacting with a multimodal interface, for example by means of speech, gestures or gaze. Therefore, correlated input events generally happen within relatively small time windows of less than 10 seconds. Synchronicity and time ordering of modalities is modelled via the CARE property-based `<par_or>`, `<par_and>`, `<seq_or>` and `<seq_and>` elements in SMUIML. These elements are in turn used to create the list of expected output sequences for all HMMs. For example, a `<transition>` declared as equivalent means that any of the modalities in the transition will lead to the same next `<context>`.

Finally, each HMM undergoes a pre-training stage, with automatically created sequences of observations injected into their applicable HMM. The Baum-Welch algorithm is used to find the unknown parameters of the HMM, hence allowing the fusion algorithm to be directly used (step 4). Basically, the internal structure of an HMM trained this way corresponds to the weighted state machine representation of the human-machine dialogue defined in the SMUIML script. The HMM can then stay this way, as a "luxury" weighted state machine, or be refined with real training data or user feedback at runtime. The HMM can also be trained at instantiation time with real training data, if it is available.

The particular strength of HMMs for the implementation of this human-machine dialogue in comparison to ad-hoc methods lies in the possibility to adapt the HMM behaviour at runtime. As pointed out in the introduction of this subsection, the main goal when creating an HMM-based fusion algorithm was to lay the foundations in HephaisTK for adding the ability to adapt to context and user feedback. HephaisTK was therefore modified to support a simple *"I am not ok with that result"* feedback from the user. If no particular result is specified, HephaisTK assumes that the last result has to be corrected. The HMM is then re-trained in order to give less weight to the erroneous result. An illustration of this feedback process is provided in the next section.

Our description of the HMM-based fusion algorithm integration is based on the SMUIML modelling language. However, the presented algorithm could be integrated into any multimodal system considering input events as normalised information atoms at the semantic level. A full modelling of the entire human-machine dialogue is not mandatory, but helps to map the interaction flow to Markov models and improves results. Also, without such a model, the training of the algorithm with help of recorded sequences would be required.

EVALUATION

The evaluation of our new multimodal fusion algorithm has been conducted in three steps. First, a qualitative evaluation using existing multimodal applications that were created based on HephaisTK has been performed. Then a more formal evaluation using the framework's integrated benchmarking tool [11] was carried out. These two tests were designed to assess the accuracy and performance of our HMM-based fusion algorithm. Third, the adaptation to user input was also tested. In addition to these three evaluations, explorations of user-induced adaptation in the context of a particular application are presented. Initial conclusions based on the results of these explorations are then discussed.

Qualitative Test Using Real-World Applications

The first task was to check whether the HMM-based algorithm aligned itself correctly with the expected results for applications that had already been developed with the HephaisTK framework based on a classic frame-based fusion algorithm. Those applications did not take into account user adaptation but were good illustrations of use cases considering the CARE-based combination of modalities. Among the tested applications, a multimodal music player application was chosen to illustrate redundancy versus equivalence cases. Another application, a tool for document management in smart meeting rooms called Docobro [12], was used for the cases of sequential and non-sequential complementarity between modalities.

Both of these two applications were migrated to the HMM-based fusion algorithm. Only a single line in the HephaisTK XML configuration file, specifying the fusion algorithm to be used, had to be changed. No other code in the SMUIML modelling script or the client Java application had to be modified. More importantly, since the SMUIML modelling script which defines the human-machine dialogues is used as preliminary training for the HMM-based fusion algorithm, no explicit training is required before the applications can be used. In fact, applications can even hot swap fusion algorithms while they are running, since context switching information is propagated in real-time to every fusion algorithm. While this was only a qualitative evaluation, the goal was to check whether the overall behaviour of the HMM-based fusion algorithm would at least be comparable to the frame-based one. Three expert users were presented with the two applications, first running with the frame-based fusion algorithm, then with the HMM-based fusion algorithm. Users were given five minutes with each of the condition to work with the application. Then an interview was conducted with them after each condition. During these interviews, users reported a coherent behaviour between both algorithms, with less false positives in the case of the HMM-based algorithm. Overall, sequenced and unsequenced complementarity, redundancy as well as equivalence cases were managed in a similar way by both algorithms. Regarding the responsiveness of the HMM-based fusion algorithm, users did not report a noticeable difference between both algorithms.

Quantitative Assessment Through an EMMA Benchmark

HephaisTK's integrated benchmarking tool [11] has been used to validate our initial observations. The benchmarking tool simulates various recognisers for a number of modalities and feeds this input data into the framework at predefined times, which have been previously recorded from a user session. The benchmarking tool then collects the fusion results in place of the client application. In the test setting shown in Figure 5, the HephaisTK framework works as if real modalities and a real client application were used.

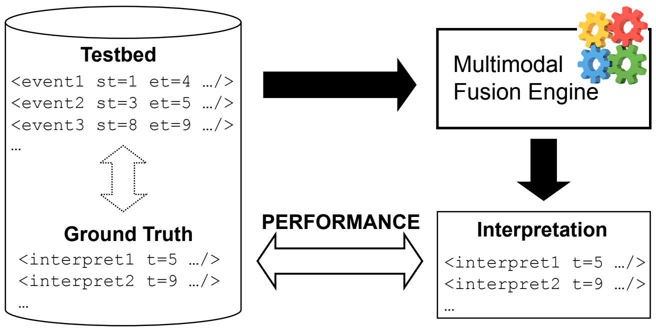


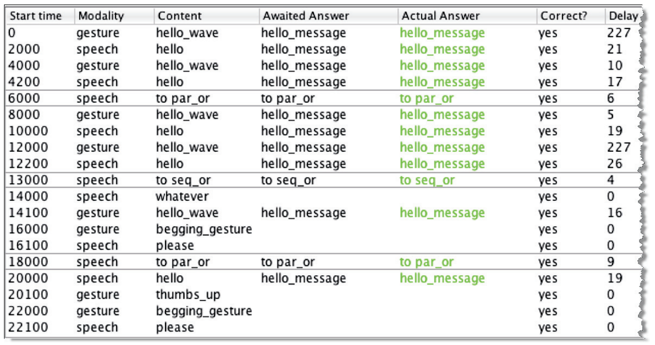
Figure 5. Benchmark test setting

The HephaisTK benchmarking tool allows for reproducible testing of the overall behaviour of the framework and of its performance and enables a comparison of different multimodal fusion algorithms. The benchmarks themselves are described via the XML EMMA language[3] which has been defined by the W3C Multimodal Interaction Working Group. Benchmarking tests were preferred over live user testing in order to compare the performance of two fusion algorithms working under the same conditions in a framework.

The results of our tests are illustrated in Figure 6 and Figure 7. Note that the screenshots of the benchmarking tool were slightly altered by adding horizontal black lines to differentiate the steps of each test. Each line corresponds to a

[3]http://www.w3.org/TR/emma/

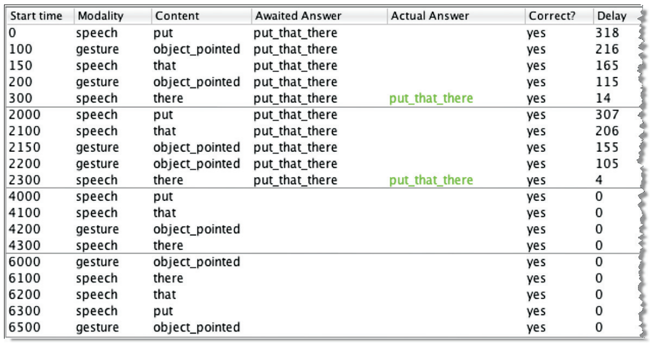
single input sent to the HephaisTK framework. Details are, from left to right: the time in milliseconds when the data was sent with 0 being the start of the test, the used modality, the content of the data and the expected answer from the fusion engine. In green or red is the answer which was effectively sent (or not) by the framework, whether this was the correct answer and finally how much time (in milliseconds) elapsed between sending the piece of data and receiving the result.



Start time	Modality	Content	Awaited Answer	Actual Answer	Correct?	Delay
0	gesture	hello_wave	hello_message	hello_message	yes	227
2000	speech	hello	hello_message	hello_message	yes	21
4000	gesture	hello_wave	hello_message	hello_message	yes	10
4200	speech	hello	hello_message	hello_message	yes	17
6000	speech	to par_or	to par_or	to par_or	yes	6
8000	gesture	hello_wave	hello_message	hello_message	yes	5
10000	speech	hello	hello_message	hello_message	yes	19
12000	gesture	hello_wave	hello_message	hello_message	yes	227
12200	speech	hello	hello_message	hello_message	yes	26
13000	speech	to seq_or	to seq_or	to seq_or	yes	4
14000	speech	whatever			yes	0
14100	gesture	hello_wave	hello_message	hello_message	yes	16
16000	gesture	begging_gesture			yes	0
16100	speech	please			yes	0
18000	speech	to par_or	to par_or	to par_or	yes	9
20000	speech	hello	hello_message	hello_message	yes	19
20100	gesture	thumbs_up			yes	0
22000	gesture	begging_gesture			yes	0
22100	speech	please			yes	0

Figure 6. Equivalence and redundancy tests

Note that some delays are well above 100 ms. These delays represent in fact the delay between the time when the first single modality data of a complex command was fed into the framework and the time when the final multimodal command was fused. For example, the *"put"* command on the first line of Figure 7 was sent to the framework 318 ms before the complete *"put that there"* command was fused.



Start time	Modality	Content	Awaited Answer	Actual Answer	Correct?	Delay
0	speech	put	put_that_there		yes	318
100	gesture	object_pointed	put_that_there		yes	216
150	speech	that	put_that_there		yes	165
200	gesture	object_pointed	put_that_there		yes	115
300	speech	there	put_that_there	put_that_there	yes	14
2000	speech	put	put_that_there		yes	307
2100	speech	that	put_that_there		yes	206
2150	gesture	object_pointed	put_that_there		yes	155
2200	gesture	object_pointed	put_that_there		yes	105
2300	speech	there	put_that_there	put_that_there	yes	4
4000	speech	put			yes	0
4100	speech	that			yes	0
4200	gesture	object_pointed			yes	0
4300	speech	there			yes	0
6000	gesture	object_pointed			yes	0
6100	speech	there			yes	0
6200	speech	that			yes	0
6300	speech	put			yes	0
6500	gesture	object_pointed			yes	0

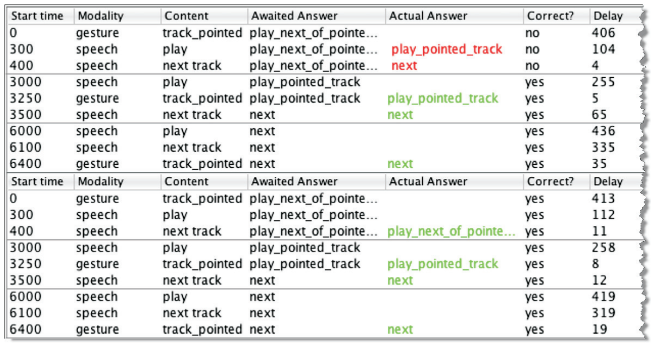
Figure 7. Sequential and non-sequential complementarity tests

The CARE properties were used as a basis to test the accuracy of the HMM-based fusion algorithm. Each CARE property was first modelled with a correct case and then with some noise introduced. Finally, erroneous commands were fed into the system to check its resistance against false positives. Figure 6 shows the results for the equivalence and redundancy cases. *"Hello"* messages were input in different modalities, first sequentially and then at the same time. The test was executed in four sequences with an increasing number of noise along the legitimate data. As shown in Figure 6, the HMM algorithm managed to pass all tests with or without added noise. The second part of this series of tests was to check whether sequential and non-sequential complementarity fusion was correctly managed by the fusion algorithm. The results are shown in Figure 7. The well-known *"put that there"* case was used to model these tests. Listing 1 shows the modelling of the command in SMUIML with parallel and sequential temporal constraints used. Four variants of the command were used: two legit variants, then an incomplete command and finally a garbled command. As outlined in Figure 7, the two correct commands were successfully fused while the incomplete and garbled commands were both rejected by the algorithm.

Listing 1. *"Put that there"* expressed in SMUIML.

```
<transition leadtime="1500">
    <seq_and>
       <trigger name="put_trigger" />
          <par_and>
             <trigger name="that_trigger" />
             <trigger name="object_pointed_event" />
          </par_and>
          <par_and>
             <trigger name="there_trigger" />
             <trigger name="object_pointed_event" />
          </par_and>
    </seq_and>
    <result action="put_that_there_action" />
</transition>
```

Finally, a challenging multimodal fusion task was tested in the form of the *"play next track"* example discussed in [11]. This task involves the differentiation between three different nuances of meaning, only based on the order of the input events. In [11], the frame-based fusion algorithm was used and could not correctly differentiate the various cases, because frames cannot easily represent this level of temporal nuances. There was a slightly better behaviour when sequential constraints were taken into account.



Start time	Modality	Content	Awaited Answer	Actual Answer	Correct?	Delay
0	gesture	track_pointed	play_next_of_pointe...		no	406
300	speech	play	play_next_of_pointe...	play_pointed_track	no	104
400	speech	next track	play_next_of_pointe...	next	no	4
3000	speech	play	play_pointed_track		yes	255
3250	gesture	track_pointed	play_pointed_track	play_pointed_track	yes	5
3500	speech	next track	next	next	yes	65
6000	speech	play	next		yes	436
6100	speech	next track	next		yes	335
6400	gesture	track_pointed	next	next	yes	35

Start time	Modality	Content	Awaited Answer	Actual Answer	Correct?	Delay
0	gesture	track_pointed	play_next_of_pointe...		yes	413
300	speech	play	play_next_of_pointe...		yes	112
400	speech	next track	play_next_of_pointe...	play_next_of_pointe...	yes	11
3000	speech	play	play_pointed_track		yes	258
3250	gesture	track_pointed	play_pointed_track	play_pointed_track	yes	8
3500	speech	next track	next	next	yes	12
6000	speech	play	next		yes	419
6100	speech	next track	next		yes	319
6400	gesture	track_pointed	next	next	yes	19

Figure 8. *"Play next track"* example without sequential constraints in the upper part and with sequential constraints in the lower part

Figure 8 shows the same tests but this time with the new HMM-based fusion algorithm. The first case without sequential constraints is still not completely solved but at least provides a consistent answer compared to most results returned by frames. The two other cases are correctly fused. As for the test run with sequential constraints, the HMM-based algorithm scores perfectly. We think that these challenging cases of ambiguous input sequences emphasise the advantage HMMs have over other classes of fusion algorithms, due to their ability to model time-related processes. However, we think that a full user evaluation would be needed in order to further assess the effects on usability of the different algorithms since our evaluation primarily focussed on the accuracy of recognition.

Performance

Multimodal interfaces should be responsive and therefore the fusion algorithms have to demonstrate reasonable performance. A full test run using the same list of commands as the tests of Figure 6 and Figure 7 was used. In total, the test run fed 40 different pieces of information into the system and expected 20 different fusion results. The full test was run 5 times with each algorithm and the delays between the input of data and the retrieval of the corresponding results were measured. 100 different measures were thus taken for each algorithm. The average computation time over these 100 measures for the frame-based fusion algorithm was 18.2 ms, with a standard deviation of 12.7 ms. On the other hand, the average computation time for HMM-based fusion algorithm was 16.6 ms, with a standard deviation of 11.6 ms, which is not significant. Further examination showed that the actual computation time of the fusion algorithms is much lower than these average values and a significant amount of the time is used by other HephaisTK computations. In fact, it appeared that most of HephaisTK's computation time is devoted to information storage and retrieval in the internal database as well as the information passing between the different software agents. In the future, we therefore intend to also test the algorithms with more complex situations and events.

User-Induced Adaptation

As presented in the related work section, adaptation can take a multitude of shapes. We are particularly interested in what is called user-induced automatic adaptation in Malinowski et al.'s framework [23]: Initiative from the user and Proposal-Decision-Execution from the machine. In this form of adaptation, the user signals to the machine that a given output was not what they expected and implicitly asks the machine to accordingly revise its judgement. The identification of the "wrong" result, the decision about a correction as well as the execution of the necessary changes are left to the machine and only the error detection part depends on the user as shown in Figure 9. The user is also not explicitly asked to select which result they would have preferred which guarantees minimal intrusion. Of course such a semi-automatic adaptation introduces some challenges for the machine processing.

In HephaisTK, the definition of the human-machine dialogue through SMUIML and the HMM-based fusion algorithm allows to explore user-induced automatic adaptation. User feedback was experimentally tested in the following way: vibration sensors were attached to the computer screen of a machine running the HephaisTK framework. When the computer screen is gently slapped, the vibration sensors are activated and provide input to the framework. Those vibration sensors events are tagged in the dialogue manager as user feedback expressing dissatisfaction from the user. The correction is sent to the HephaisTK `FusionManager` which adapts the behaviour of the HMM-based fusion algorithm accordingly. The erroneous result was in our test case simply considered to be the last output produced by the fusion algorithm. The HMM for the current context is trained to give less weight to the particular transition which triggered this erroneous result.

We have conducted some tests of this setting with help of the benchmarking tool, in order to verify the correct behaviour of the algorithm. The same tests as explained above were used, but in the case of a wrongly recognised result, simulated user feedback was injected into HephaisTK. Three different cases were tested and in all three cases the algorithm adapted successfully its behaviour. The same setting was then used "live" with the Docobro document browser and the multimodal music player.

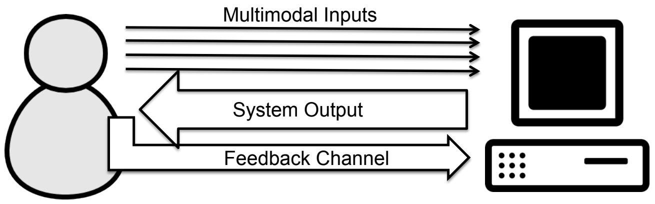


Figure 9. User-based error detection and feedback

Based on our tests, we can draw the conclusion that the integration of user-induced automatic adaptation in multimodal interfaces raises the following challenges: the identification of the problematic fusion result, the identification of the source of the error and the creation of a meaningful correction as well as the possibility for undo operations.

When the user indicates the wish for a fusion result to be corrected, other fusion results might already have been processed before the user's feedback reaches the fusion engine. In the presented test environment, the erroneous fusion result was always considered to be the last one. Evidently, this strategy proved soon to be error prone. Since the HMM-based fusion algorithm provides a probability for the correct recognition with each fusion result, this probability score was used to try to give less weight to the least probable fusion result among the most recent ones. This strategy was not satisfactory either and the identification of the erroneous fusion behaviour among the latest results without further input from the user is still an open issue.

The identification of the source of error is also an open issue. Indeed, the fusion algorithm works with interpretations coming from different recognisers. If the erroneous fusion behaviour originates from a recognition mistake of a particular modality (e.g. speech recogniser), downgrading a fusion result at the fusion algorithm level could worsen the fusion results rather than improving them. On the other hand, if the source of the error is correctly identified, the individual modality recognisers have the possibility to make use of a user's feedback to correct their results. We assume that extensive testing could help to identify thresholds at which "bad" results should be delegated either to the fusion algorithm or to individual recognisers, or be simply ignored if no source could be assigned.

The meaningful correction was satisfactory throughout the tests achieved with the previously described setting. When incorrect results originating from the latest fusion result were produced by the HMM-based fusion algorithm, a retraining of the current context's HMM lead to a reduction of false positives. However, if the error did not originate from the latest fusion algorithm result, the lack of some "undo" functionality showed to be problematic. Keeping two copies of each HMM and having one version trained one step later than the other should help to introduce this "undo" mechanism at the technical level.

CONCLUSION

We presented our work on a novel algorithm for decision-level fusion of input data in the context of multimodal interactive systems. Our algorithm is based on hidden Markov models and has been implemented and tested in the HephaisTK framework. A qualitative as well as quantitative assessment of the algorithm has been conducted with HephaisTK's integrated benchmarking tool. We conducted repeated tests of two multimodal fusion algorithms by feeding them with multimodal inputs organised in a series of examples. These examples represented different cases of unambiguous and ambiguous time-synchronised multimodal input. With similar processing time, our new HMM-based fusion algorithm demonstrated similar or better accuracy than a classic frame-based fusion algorithm. This improvement in accuracy, coupled with the capability of our new HMM-based algorithm to take temporal combinations of modalities into account, makes it a promising solution for decision-level fusion engines of interactive multimodal systems. Furthermore, the algorithm demonstrated its strength in adapting fusion results based on user and context information. Our initial explorations on automatic user-induced adaptation in multimodal interfaces revealed three main challenges to be overcome before the presented type of adaptivity can be fully employed in multimodal interfaces: 1) the identification of the problematic fusion result, 2) the identification of the source of error and 3) error correction and learning.

In the future, we plan to explore automatic user-induced adaptation, based on the challenges identified in this paper. We will do further evaluations through tests on specific tasks and hypotheses, as well as a user study with more users. We further plan to focus on making full use of probability scores provided by individual modality recognisers and go beyond the simple integration in HMMs. The robust handling of inputs with uncertainty needs to be tackled not only on the algorithm level but also on the dialogue level. To this end, we plan to integrate a variant of the framework proposed by Schwarz et al. [27] into HephaisTK, which should complement the management of the probability scores at the algorithm level.

ACKNOWLEDGMENTS

The work on HephaisTK and SMUIML has been funded by the Hasler Foundation in the context of the MeModules project and by the Swiss National Center of Competence in Research on Interactive Multimodal Information Management via the NCCR IM2 project. Bruno Dumas is supported by MobiCraNT, a project forming part of the Strategic Platforms programme by the Brussels Institute for Research and Innovation (Innoviris).

REFERENCES

1. Arhippainen, L., Rantakokko, T., and Tähti, M. Navigation with an Adaptive Mobile Map-Application: User Experiences of Gesture- and Context-Sensitiveness. In *Proceedings of the 2nd International Conference on Ubiquitous Computing Systems (UCS 2004)* (Tokyo, Japan, November 2005), 62–73.
2. Baum, L., Petrie, T., Soules, G., and Weiss, N. A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains. *The Annals of Mathematical Statistics 41*, 1 (1970), 164–171.
3. Bolt, R. A. "Put-that-there": Voice and Gesture at the Graphics Interface. In *Proceedings of the 7th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 1980)* (Seattle, USA, July 1980), 262–270.
4. Bourguet, M.-L. A Toolkit for Creating and Testing Multimodal Interface Designs. In *Companion of the 15th Annual Symposium on User Interface Software and Technology (UIST 2002)* (Paris, France, October 2002), 29–30.
5. Cesar, P., Vaishnavi, I., Kernchen, R., Meissner, S., Hesselman, C., Boussard, M., Spedalieri, A., Bulterman, D. C., and Gao, B. Multimedia Adaptation in Ubiquitous Environments: Benefits of Structured Multimedia Documents. In *Proceedings of the 8th ACM Symposium on Document Engineering (DocEng 2008)* (São Paulo, Brazil, September 2008), 275–284.
6. Chai, J., Hong, P., and Zhou, M. A Probabilistic Approach to Reference Resolution in Multimodal User Interfaces. In *Proceedings of the 9th International Conference on Intelligent User Interfaces (IUI 2004)* (Funchal, Madeira, Portugal, January 2004), 70–77.
7. Chen, Q., Georganas, N. D., and Petriu, E. M. Hand Gesture Recognition Using Haar-Like Features and a Stochastic Context-Free Grammar. *IEEE Transactions on Instrumentation and Measurement 57*, 8 (August 2008), 1562–1571.
8. Choudhury, T., Clarkson, B., Jebara, T., and Pentland, A. Multimodal Person Recognition using Unconstrained Audio and Video. In *Proceedings of the International Conference on Audio- and Video-Based Person Authentication (AVBPA 1999)* (Washington DC, USA, 1999), 176–181.
9. Cohen, P. R., Johnston, M., McGee, D., Oviatt, S., Pittman, J., Smith, I., Chen, L., and Clow, J. QuickSet: Multimodal Interaction for Simulation Set-Up and Control. In *Proceedings of the 5th Conference on Applied Natural Language Processing (ANLC 1997)* (Washington DC, USA, April 1997), 20–24.

10. Coutaz, J., Nigay, L., Salber, D., Blandford, A., May, J., and Young, R. M. Four Easy Pieces for Assessing the Usability of Multimodal Interaction: The CARE Properties. In *Proceedings of the 5th International Conference on Human-Computer Interaction (Interact 1995)* (Lillehammer, Norway, June 1995), 115–120.

11. Dumas, B., Ingold, R., and Lalanne, D. Benchmarking Fusion Engines of Multimodal Interactive Systems. In *Proceedings of the 11th International Conference on Multimodal Interfaces (ICMI-MLMI 2009)* (Cambridge, USA, November 2009), 169–176.

12. Dumas, B., Lalanne, D., and Ingold, R. HephaisTK: A Toolkit for Rapid Prototyping of Multimodal Interfaces. In *Proceedings of the 11th International Conference on Multimodal Interfaces (ICMI-MLMI 2009)* (Cambridge, USA, November 2009), 231–232.

13. Dumas, B., Lalanne, D., and Ingold, R. Description Languages for Multimodal Interaction: A Set of Guidelines and its Illustration with SMUIML. *Journal on Multimodal User Interfaces: "Special Issue on The Challenges of Engineering Multimodal Interaction" 3*, 3 (February 2010), 237–247.

14. Flippo, F., Krebs, A., and Marsic, I. A Framework for Rapid Development of Multimodal Interfaces. In *Proceedings of the 5th International Conference on Multimodal Interfaces (ICMI 2003)* (Vancouver, Canada, November 2003), 109–116.

15. Forney Jr, G. The Viterbi Algorithm. *Proceedings of the IEEE 61*, 3 (1973), 268–278.

16. Hoste, L., Dumas, B., and Signer, B. Mudra: A Unified Multimodal Interaction Framework. In *Proceedings of the 13th International Conference on Multimodal Interfaces (ICMI 2011)* (Alicante, Spain, November 2011), 97–104.

17. Huang, J., Liu, Z., Wang, Y., Chen, Y., and Wong, E. Integration of Multimodal Features for Video Scene Classification Based on HMM. In *Proceedings of the 3rd IEEE Workshop on Multimedia Signal Processing (MMSP 1999)* (Copenhagen, Denmark, September 1999), 53–58.

18. Johnston, M., Cohen, P., McGee, D., Oviatt, S., Pittman, J., and Smith, I. Unification-Based Multimodal Integration. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL 1997)* (Madrid, Spain, July 1997), 281–288.

19. Juang, B., and Rabiner, L. Hidden Markov Models for Speech Recognition. *Technometrics 33* (1991), 251–272.

20. König, W. A., Rädle, R., and Reiterer, H. Squidy: A Zoomable Design Environment for Natural User Interfaces. In *Proceedings of the 27th International Conference on Human Factors in Computing Systems (CHI 2009)* (Boston, USA, April 2009), 4561–4566.

21. Lalanne, D., Nigay, L., Palanque, P., Robinson, P., Vanderdonckt, J., and Ladry, J. Fusion Engines for Multimodal Input: A Survey. In *Proceedings of the 11th International Conference on Multimodal Interfaces (ICMI-MLMI 2009)* (Cambridge, USA, September 2009), 153–160.

22. López-Jaquero, V., Vanderdonckt, J., Montero, F., and González, P. Towards an Extended Model of User Interface Adaptation: The Isatine Framework. In *Engineering Interactive Systems*, J. Gulliksen, M. B. Harning, P. Palanque, G. C. Veer, and J. Wesson, Eds., vol. 4940 of *LNCS*. Springer Verlag, 2008, 374–392.

23. Malinowski, U., Thomas, K., Dieterich, H., and Schneider-Hufschmidt, M. A Taxonomy of Adaptive User Interfaces. In *Proceedings of the Conference on People and Computers VII (HCI 1992)* (York, United Kingdom, September 1992), 391–414.

24. Octavia, J., Raymaekers, C., and Coninx, K. Adaptation in Virtual Environments: Conceptual Framework and User Models. *Multimedia Tools and Applications 54* (2011), 121–142.

25. Oviatt, S. Human-Centered Design Meets Cognitive Load Theory: Designing Interfaces That Help People Think. In *Proceedings of the 14th ACM International Conference on Multimedia (ACM MM 2006)* (Santa Barbara, USA, October 2006), 871–880.

26. Rabiner, L. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE 77*, 2 (1989), 257–286.

27. Schwarz, J., Hudson, S., Mankoff, J., and Wilson, A. D. A Framework for Robust and Flexible Handling of Inputs with Uncertainty. In *Proceedings of the 23nd Symposium on User Interface Software and Technology (UIST 2010)* (New York, USA, October 2010), 47–56.

28. Serrano, M., Nigay, L., Lawson, J., Ramsay, A., Murray-Smith, R., and Denef, S. The OpenInterface Framework: A Tool for Multimodal Interaction. In *Proceedings of the 26th International Conference on Human Factors in Computing Systems (CHI 2008)* (Florence, Italy, April 2008), 3501–3506.

29. Sharma, R., Pavlovic, V., and Huang, T. Toward Multimodal Human-Computer Interface. *Proceedings of the IEEE 86*, 5 (1998), 853–869.

30. Vo, M., and Wood, C. Building an Application Framework for Speech and Pen Input Integration in Multimodal Learning Interfaces. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 1996)* (Atlanta, USA, May 1996), 3545–3548.

31. Wu, L., Oviatt, S., and Cohen, P. From Members to Teams to Committee – A Robust Approach to Gestural and Multimodal Recognition. *IEEE Transactions on Neural Networks 13*, 4 (2002), 972–982.