Mudra: A Unified Multimodal Interaction Framework

Lode Hoste, Bruno Dumas and Beat Signer
Web & Information Systems Engineering Lab
Vrije Universiteit Brussel
Pleinlaan 2, 1050 Brussels, Belgium
{lhoste,bdumas,bsigner}@vub.ac.be

ABSTRACT

In recent years, multimodal interfaces have gained momentum as an alternative to traditional WIMP interaction styles. Existing multimodal fusion engines and frameworks range from low-level data stream-oriented approaches to high-level semantic inference-based solutions. However, there is a lack of multimodal interaction engines offering native fusion support across different levels of abstractions to fully exploit the power of multimodal interactions. We present Mudra, a unified multimodal interaction framework supporting the integrated processing of low-level data streams as well as high-level semantic inferences. Our solution is based on a central fact base in combination with a declarative rule-based language to derive new facts at different abstraction levels. Our innovative architecture for multimodal interaction encourages the use of software engineering principles such as modularisation and composition to support a growing set of input modalities as well as to enable the integration of existing or novel multimodal fusion engines.

Keywords

multimodal interaction, multimodal fusion, rule language, declarative programming

Categories and Subject Descriptors

D.2.11 [**Software Engineering**]: Software Architectures; H.5.2 [**Information Interfaces and Presentation**]: User Interfaces

General Terms

Algorithms, Languages

1. INTRODUCTION

Multimodal interaction and interfaces have become a major research topic over the last two decades, representing a new class of user-machine interfaces that are different from standard WIMP interfaces. As stated by Oviatt [13], multimodal interfaces have the added capability to process multiple user input modes not only in a parallel manner, but also by taking temporal and semantic combinations between different input modes into account. These interfaces tend to emphasise the use of richer and more natural ways of communication, including speech or gesture, and more generally address all five senses. Hence, the objective of multimodal interfaces is twofold: first, to support and accommodate a user's perceptual and communicative capabilities; and second, to embed computational power in the real world, by offering more natural ways of human-computer interaction [5].

However, the process of recovering the user intent through multiple different input sources and their potential combination, known as "multimodal input fusion", presents a number of challenges to be overcome before multimodal interfaces can be experienced to their fullest. First, the processing has to happen in real time, demanding for architectures to efficiently manage parallel input streams as well as to perform the recognition and fusion in the presence of temporal constraints. Second, the type of data to be managed by a multimodal system may originate from a variety of different sources. For example, a multi-touch surface might deliver multiple streams of pointer positions along with identifiers for different fingers, hands or even different users. On the other hand, a speech recogniser may deliver a list of potential text results in combination with the corresponding recognition probabilities. Being able to fuse user input from such different channels is one of the strengths of multimodal interfaces, but in practice only a few tools have been able to fully support data-agnostic input. In effect, most multimodal interaction tools either focus on extracting semantic interpretations out of the input data or offer low-level management of input data streams. Semantic-level tools are typically high-level frameworks that consume semantic events from multiple modalities to achieve a more intuitive and improved human-computer interaction. In these approaches, fusion is done on sets of individual high-level interpretations mostly coming from different recognisers. The second family of tools and frameworks focus strongly on dataflow paradigms to process raw data at the data level. These approaches typically declare the flow of primitive events by chaining multiple *boxes* serving as a filter or via event-specific fusion of multiple sources. The consequence is that we have, on the one hand, semantic tools which struggle with low-level and high frequency data and, on the other hand, frameworks to manage low-level input data streams which have to resort to ad hoc and case by case implementations for higher-level information fusion.

Sharma et al. [17] identified three different levels of fusion of input data: *data-level fusion*, *feature-level fusion* and *decision-level fusion*. These fusion levels are seen as distinct entities, working at completely different stages. On the other hand, fusion as a whole is supposed to be able to take into account data from any of these three levels, be it x/y coordinates from a pointer or semantic information

ICMI'11, November 14–18, 2011, Alicante, Spain.

coming from speech. In this paper, we address this conflict between seemingly irreconcilable fusion levels and data-agnostic input. A first open issue is the correlation between the events processed by dataflow approaches with a high throughput and low-throughput high-level events produced by semantic inference-based fusion. A second issue is how to retain access to low-level information when dealing with interpreted high-level information. For example, a gesture can be described by a set of x/y coordinates, by a sequence of atomic time-stamped vectors or by a semantic interpretation denoting that the user has drawn an upward-pointing arrow. Only a few tools keep track of the three different information levels and consider that users might express either deictic pointing information, manipulative movements or iconic gestures. Finally, in order to achieve multimodal fusion, some specific metadata has to be extracted regardless of the level of fusion. For instance, for raw-level data provided by a simple deictic pointing gesture that is freely performed and captured by a 3D camera, the start and end time of the gesture is required in order to resolve temporal relationships with other modalities.

To accommodate this missing link between low-level and high-level events, we developed a unified multimodal fusion engine capable of reasoning over both primitive and high-level information based on time window constructs. We encourage the use of modularisation and composition to build reusable and easily understandable building blocks for multimodal fusion. These software principles are emphasised by Lalanne et al. [11] stating that the *"engineering aspects of fusion engines must be further studied, including the genericity (i.e., engine independent of the combined modalities), software tools for the fine-tuning of fusion by the designer or by the end-users as well as tools for rapidly simulating and configuring fusion engines to a particular application by the designer or by the end-users."*

We start by discussing related work in Section 2 and investigate how existing approaches address the correlation between low-level and high-level events. In Section 3, we present the architecture of our multimodal interaction framework, called Mudra, and introduce the features that enable Mudra to deal with data-level, feature-level as well as decision-level fusion while retaining temporary fusion information. We further introduce the declarative rule-based language forming part of the Mudra core. A discussion and comparison of our unified multimodal interaction framework with existing multimodal engines as well as potential future directions are provided in Section 4. Concluding remarks are given in Section 5.

2. BACKGROUND

The fusion of multimodal input data can take place at different levels of abstraction. In this section, we first present these different levels together with some classical use cases. We then discuss existing solutions for the fusion of multimodal input at these abstraction levels and identify some of their limitations.

2.1 Multimodal Fusion Levels

As mentioned earlier, Sharma et al. [17] distinguish three levels of abstraction to characterise multimodal input data fusion: data-level fusion, feature-level fusion and decision-level fusion.

- *Data-level fusion* focuses on the fusion of identical or tightly linked types of multimodal data. The classical illustration of data-level fusion is the fusion of two video streams coming from two cameras filming the same scene at different angles in order to extract the depth map of the scene. Data-level fusion rarely deals with the semantics of the data but tries to enrich or correlate data that is potentially going to be processed by higher-level fusion processes. As data-level fusion works on the raw data, it has access to the detailed information but is also highly sensitive to noise or failures. Data-level fusion frequently entails some initial processing of raw data including noise filtering or very basic recognition.

- *Feature-level fusion* is one step higher in abstraction than data-level fusion. Typically, data has already been processed by filters and fusion is applied on features extracted from the data rather than on the raw data itself. Feature-level fusion of modalities typically applies to closely coupled modalities with possibly different representations. A classical example is speech and lip movement integration [14], where data comes from a microphone that is recording speech as well as from a camera filming the lip movements. The two data streams are synchronised and in this case the goal of the data fusion is to improve speech recognition by combining information from the two different modalities. Feature-level fusion is less sensitive to noise or failures than data-level fusion and conveys a moderate level of information detail. Typical feature-level fusion algorithms include statistical analysis tools such as Hidden Markov Models (HMM), Neural Networks (NN) or Dynamic Time Warping (DTW).

- *Decision-level fusion* is centered around deriving interpretations based on semantic information. It is the most versatile kind of multimodal fusion, as it can correlate information coming from loosely coupled modalities, such as speech and gestures. Decision-level fusion includes the merging of high-level information obtained by data- and feature-level fusion as well as the modelling of human-computer dialogues. Additionally, partial semantic information originating from the feature level can yield to mutual disambiguation [12]. Decision-level fusion is assumed to be highly resistant to noise and failures as it relies on the quality of previous processing steps. Therefore, the information that is available for decision-level fusion algorithms may be incomplete or distorted. Typical classes of decision-level fusion algorithms are meaning frames, unification-based or symbolic-statistical fusion algorithms.

Note that a single modality can be processed on all three fusion levels. For example, speech can be processed at the signal (data) level, phonemes (features) level or utterances (decision) level. In the case of speech, the higher fusion levels might use results from lower-level fusion. Surprisingly, existing multimodal interaction frameworks often excel at one specific fusion level but encounter major difficulties at other levels. We argue that the reason for these limitations lies on the architecture level and in particular how the initial data from different modalities is handled.

2.2 Data Stream-Oriented Architecture

One approach to build multimodal interaction architectures is to assume a continuous stream of information coming from different modalities and to process them via a number of chained filters. This is typically done to efficiently process streams of high frequency data and to perform fusion on the data and/or feature level. Representatives of this strategy are OpenInterface [16] and Squidy [10], employing a data stream-oriented architecture to process raw data sources and fuse multiple sources on an event-per-event basis.

Although these data stream approaches advocate the use of *composition boxes*, they do not provide a fundamental solution to define temporal relations between multiple input sources. All incoming events are handled one by one and the programmer manually needs

to take care of the intermediate results. This leads to a difficult management of complex semantic interpretations. Data stream-oriented architectures show their limits when high-throughput information such as accelerometer data (i.e. more than 25 events per second) should be linked with low-throughput semantic-level information such as speech (i.e. less than one event per second). When confronted with the fusion of information coming from different abstraction levels, these architectures tend to rely on a case-by-case approach, thereby losing their genericity. Furthermore, the decision-level fusion of semantic information between multiple modalities requires classes of algorithms, such as meaning frames, which address temporal relationships and therefore need some kind of intermediate storage (e.g. *registers*). These algorithms are not in line with the stream-oriented architecture and developers have to rely on ad hoc solutions.

2.3 Semantic Inference-Based Approach

A second type of architecture for multimodal interaction focuses on supporting fusion of high-level information on the decision level. These approaches offer constructs to specify sets of required information before an action is triggered. Information gathered from the different input modalities is assumed to be classified correctly. Furthermore, these approaches work best with relatively low frequency data and highly abstracted modalities.

Four classes of fusion algorithms are used to perform decision-level fusion:

- Meaning frame-based fusion [19] uses data structures called frames for the representation of semantic-level data coming from various sources or modalities. In these structures, objects are represented as attribute/value pairs.
- Unification-based fusion [9] is based on recursively merging attribute/value structures to infer a high-level interpretation of user input.
- Finite state machine-based approaches [8] model the flow of input and output through a number *states*, resulting in a better integration with strongly temporal modalities such as speech.
- Symbolic/statistical fusion, such as the Member-Team-Committee (MTC) algorithm used in Quickset [21] or the probabilistic approach of Chai et al. [2], 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.

The presented approaches work well for the fusion of semantic-level events. However, when confronted with lower level data, such as streams of 2D/3D coordinates or data coming from accelerometers, semantic inference-based approaches encounter difficulties in managing the high frequency of input data. In order to show their potential, these approaches assume that the different modalities have already been processed and that we are dealing with semantic-level information.

However, even when confronted with semantic-level data, several issues can arise with existing approaches. First, they have to fully rely on the results of the modality-level recognisers without having the possibility to exploit the raw information at all. This can lead to problems in interpretation, for example with continuous gestures (e.g. pointing) in thin air.

Second, as decision-level fusion engines assume that the creation of semantic events happens at a lower level, they have no or only limited control over the refresh rate of these continuous gestures. The typical solution for this scenario is to create a single pointing event at the time the hand was steady. Unfortunately, this considerably slows down the interaction and introduces some usability issues. Another approach is to match the pointing gesture for every time step on the discrete time axis; for example once per second. However, this conflicts with the occupied meaning frame slot and demands for ad hoc solutions.

A third issue that arises when employing meaning frames or similar fusion algorithms is related to the previously discussed problem. Suppose that a user aborts and restarts their interaction with the computer by reissuing their commands. Already recognised information from the first attempt, such as a "hello" speech utterance, are already occupying the corresponding slot in the meaning frame. A second triggering of "hello" will either be refused and possibly result in a misclassification due to an unexpected time span when matched with a newer pointing gesture, or it will overwrite the existing one which introduces problems for partially overlapping fusion since meaningful scenarios might be dropped.

Finite state machine-based approaches such as [8], typically lack the constructs to express advanced temporal conditions. The reason is that a finite state machine (FSM) enforces the input of events in predefined steps (i.e. event x triggers a transition from state a to b). When fusing concurrent input, all possible combinations need to be manually expressed.

The two major benefits are the flexible semantic and temporal relations between edges and the inherent support for probabilistic input. However, the manual construction of complex graphs becomes extremely difficult to cope with as the number of cases to be taken into account is growing. When such systems have to be trained, the obvious problem of collecting training sets arises and once again increases with the number of considered cases. Additionally, these approaches require a strict segmentation of the interaction. This implies a clear specification of the start and stop conditions before the matching occurs. Hence, supporting overlapping matches introduces some serious issues and also has an impact on the support for multiple users and the possible collaborative interaction between them.

2.4 Irreconcilable Approaches?

Finally, other issues, such as multi-user support, are currently problematic in both data stream-oriented and semantic inference-based approaches. For instance, at the raw data level, potentially available user information is frequently lost as data is treated at the same level as other pieces of data and requires some ad hoc implementations. When employing meaning frame-based fusion or any other decision-level fusion, slots can be occupied by any user. However, this means that events from one participant can be undesirably composed with events from another user. Note that a major additional effort is required from the programmer to support multi-user scenarios, since every meaning frame has to be manually duplicated with a constant constraint on the `user` attribute, resulting in an increased fusion description complexity.

In conclusion, data stream-oriented architectures are very efficient when handling data streams and semantic inference-based approaches process semantic-level information with ease. However, none of the presented approaches is efficient in handling both high frequency data streams at a low abstraction level and low frequency semantic pieces of information at a high abstraction level. Not to mention the possibility to use data-level, feature-level and decision-level information of the very same data stream at the same time. In the next section, we present our unified multimodal interaction architecture called Mudra, which reconciles the presented approaches by supporting fusion across the different abstraction levels.

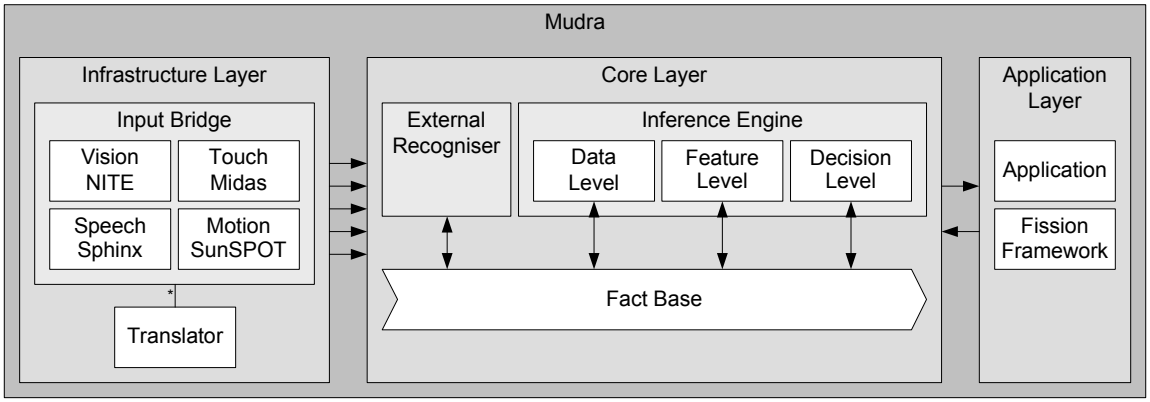

Figure 1: Mudra architecture

3. MUDRA

In order to build a fusion engine that is able to process information on the data, feature and decision level in real-time, we believe that a novel software architecture is needed. We present Mudra, our multimodal fusion framework which aims to extract meaningful information from raw data, fuse multiple feature streams and infer semantic interpretation from high-level events.

The overall architecture of the Mudra framework is shown in Figure 1. At the infrastructure level, we support the incorporation of any arbitrary input modalities. Mudra currently supports multiple modalities including skeleton tracking via Microsoft's Xbox Kinect in combination with the NITE[1] package, cross-device multi-touch information via TUIO and Midas [15], voice recognition via CMU Sphinx[2] and accelerometer data via SunSPOTs[3]. These bindings are implemented in the infrastructure layer. On arrival, event information from these modalities is converted into a uniform representation, called facts, and timestamped by the translator. A fact is specified by a type (e.g. speech) and a list of attribute/value pairs, called slots (e.g. `word` or `confidence`). For example, when a user says "put", the fact shown in Listing 1 is inserted into a fact base. Facts can address data coming from any level of abstraction or even results from fusion processing. A fact base is a managed collection of facts, similar to a traditional database.

Listing 1: "Put" event via Speech

```
1 (Speech (word "put") (confidence 0.81)
2         (user "Lode") (on 1305735400985))
```

However, instead of activating queries on demand, we use continuous rules to express conditions to which the interaction has to adhere. A production rule consists of a number of prerequisites (before the $\Rightarrow$) and one or more actions that are executed whenever the rule is triggered. Such a prerequisite can either be a fact match or a test function. A match is similar to an open slot in meaning frames but with the possibility to add additional constraints and boolean features, which leads to more flexible expressions. Test functions are user defined and typically reason over time, space or other constraints (e.g. `tSequence` or `tParallel` for sequential and parallel temporal constraints). Finally, when all prerequisites are met, an action (after the $\Rightarrow$) is triggered. Typical behaviour for an action is the assertion of a new, more meaningful fact in the fact base while bundling relevant information.

[1]NITE and OpenNI: `http://www.openni.org`
[2]CMU Sphinx: `http://cmusphinx.sourceforge.net`
[3]Sun SunSPOT: `http://www.sunspotworld.com`

The encapsulation of data enables the modularisation and composition while modelling complex interactions. This is inherently supported by our approach and allows developers to easily encode multimodal interaction. These constructs form the basis of our solution and allow developers to match the complete range from low-level to high-level events. The inference engine is based on CLIPS[4] (C Language Integrated Production System), which is an expert system tool developed by the Technology Branch of the NASA Lyndon B. Johnson Space Center. We have substantially extended this tool with an extensive infrastructure layer, the support for continuous evaluation, the inclusion of machine learning-based recognisers (e.g. DTW and HMM) and a network-based communication bus to the application layer. The application layer provides flexible handlers for end-user applications or fission frameworks, with the possibility to feed application-level entities back to the core layer.

3.1 Unified Multimodal Fusion

3.1.1 Data-Level processing

Data-level processing is primarily used for two purposes in the Mudra framework: noise filtering and recognition. Kalman filtering [20] typically allows for easier recognition of gestures in accelerometer data. This processing is achieved at the infrastructure layer since filtering is tightly coupled with specific modalities. Employing rules at the data level has already been shown to be effective for the recognition of complex multi-touch gestures [15]. The use of production rules to encode gestures based on vision data-level input has also been exploited by Sowa et al. [18] who used the following declarative encoding for a pointing gesture: *"If the index finger is stretched, and if all other fingers are rolled (pointing hand- shape) and if the hand simultaneously is far away from the body, then we have a pointing gesture"*. A similar approach is used in Mudra in the form of production rules to deal with the correlation of information at the data level.

3.1.2 Feature-Level processing

To improve recognition rates, fusion at the feature level can be used to disambiguate certain cases where a single modality falls short. For example, in multi-touch technology, every finger gets assigned a unique identifier. However this does not provide information whether these fingers originate from the same hand or from different users. The fusion of existing techniques, for example shadow images [6] or the use of small amounts of electrical current to identify individual users [4], is possible at the feature level.

[4]CLIPS: `http://clipsrules.sourceforge.net`

It is important to stress that we do not enforce a strict dataflow from the data level to the feature level. This has the advantage that data-level recognisers can benefit from information provided at the feature level. If existing feature-level techniques are incorporated in the framework, the data-level processing of multi-touch gestures can, for example, immediately profit from their results.

3.1.3 Decision-Level processing

At the decision level, the advanced modelling of multimodal fusion can be described in a very flexible manner, as developers have access to events ranging from low to high level. External data-level, feature-level or decision-level fusion algorithms can also be applied to any facts available in the fact base. The underlying complexity is hidden from developers, as illustrated in Listing 2 showing our implementation of Bolt's famous *"put that there"* example [1].

Listing 2: Bolt's "Put that there"

```
1  (defrule bolt
2       (declare (global−slot−constraint (user ?user)))
3       ?put <− (Voice (word "put") {> confidence 0.7})
4       ?that <− (Voice (word "that"))
5       ?thatp <− (Point)
6       (test (tParallel ?that ?thatp))
7       ?there <− (Voice (word "there"))
8       ?therep <− (Point)
9       (test (tParallel ?there ?therep))
10      (test (tSequence ?put ?that ?there))
11 =>
12      (assert (BoltInteraction)))
```

In this fusion example, we assume some high-level events. For example, line 3 and 4 show a pattern match on a voice fact containing a "put" (see Listing 1) and a "that" string at the `word` slot. Resulting fact matches are bound to variables, denoted by a question mark (i.e. `?put` and `?that`). Line 5 specifies a point event, which could be issued by a touch interaction or a hand pose. The point fact also contains a modality type slot, but if the developer does not constrain the attribute information to a single or multiple modalities, the rule will trigger for all cases. This example illustrates the abstraction level of our declarative rules, where the underlying complexity of the point event is hidden by one or multiple rules or external recognisers.

Different temporal operators, such as `tParallel` (line 6) or `tSequence` (line 10), are user defined rather than being fixed and limited to engine-level constructs. Developers can introduce their own operators at any time. It is important to note that the voice events generated by recognisers in the infrastructure layer are merged with point events extracted by data-level processing. Although pointing is a continuous interaction which generates multiple events per second, our system is able to fuse both inputs. Fusion algorithms at different fusion levels may find patterns in the fact base. In the future we plan to further exploit this feature by dynamically analysing speech in fusion-aware speech grammars.

3.2 Fundamental Features of Mudra

Attribute Constraints Additional constraints can be enforced by developers before a matched fact type is bound to a variable. A first constraint is realised by assigning a constant value on an attribute. This is shown on line 3 of Listing 2 by stating the string `"put"` for the `word` slot. A boolean OR operator supports alternative constant values in the case that this is required.

A second interesting constraint is available via inline function calls (denoted by curly braces), which is outlined on line 3. This construct not only supports boolean operators such as AND, OR and NOT, but it can also be used to specify value ranges (e.g. for sliders) or to call user-defined functions. In Listing 2, we applied an inline function call to enforce a minimal probability for the correctness of the recognised word.

A third type of attribute constraint is to use variable bindings. In production rules, a variable can only be bound once. Thus, if a single variable is used at multiple locations, it indicates that all these instances should contain the same value. This is very flexible as developers are not enforced to provide a constant value. Typically, this feature is applied in a multi-user context, where all matched events should be produced by a user without referring to a particular username. In Listing 2, we applied this mechanism via a macro (`global-slot-constraint`) which spans over all fact matches that contain user information.

Negation of Events A powerful feature is the use of negation to denote that an event should not happen during the defined scenario. This construct can also be used to define priorities between different modalities, for instance to express that pointing should be active as long as there is no voice input. The rich expressiveness of the negation feature is very handy when describing certain types of multimodal interactions.

Local Integration of Probability Probability information originating from external recognisers (e.g. speech recognition) can be integrated as attribute values. Due to the advanced attribute constraint mechanism, a threshold can be set locally and is not required to be system wide. It is very interesting to exploit this feature to reduce false positives as one can enforce a higher threshold for key components of the fusion. For instance, line 3 in Listing 2 requires a recognition probability higher than 0.7 for the speech recognition of "put", which is higher than the default threshold of 0.5. When extending the system with additional but similar fusion rules like "clone that there", the possibility to refine these probabilities on a per-event basis is a clear advantage.

Overlapping Matches Support for overlapping matches is an important benefit of our approach and enables to bridge the gap between low-level and high-level fusion. New events that overlap with partial matches are not thrown away but create new, additional partial matches. The mechanism relates to an automated replication strategy of meaning frames whenever a register is occupied. Developers do not have to decide between skipping new events or overwriting existing partial matches. Overlapping matches are handled very efficiently by the Rete algorithm [7]. Since we inherently support the bookkeeping of partial matches, we provide an additional `delay` construct to control the frequency of the rule triggering. The delayed triggering is important for data-level processing since many low-level events correlate to similar conclusions and a reduction of events minimises the processing necessary by the inference engine. Note that the delay construct can be applied to define the refresh rate of continuous gestures (e.g. pointing). Decision-level fusion greatly benefits from this control mechanism in our unified multimodal framework.

Sliding Window The fact base only contains facts that have not yet outlived their time span. This time span parameter is necessary for performance and memory reasons. A time span is specified per fact type, which allows developers to keep high-level semantic events with a low throughput longer in the fact base than low-level events generated with a higher refresh rate. The result is a flexible time-windowing strategy where developers can choose between performance and accessibility of older data for fusion.

Multi-User Support Multi-user support is exploited by specifying conditions on attributes. Whenever any user information is available—either originating from hardware, extracted by a recogniser or fused from multiple modalities—it can be included as an attribute in a fact. As mentioned earlier, the use of a single variable binding in a rule can be used to enforce events generated by the same user. However, specifying this attribute for every conditional element introduces a lot of redundant program code resulting in more complex rules. We therefore introduced a new language construct to declare constraints on all matches whenever the specified attribute is present. This is illustrated on line 2 of Listing 2 which enforces all events to be issued by the same user. Due to the inherent support for overlapping matches, Listing 2 supports the concurrent interaction of multiple users in the multimodal "put that there" scenario.

Collaborative User Support To go one step further, we show how rules can be employed to support collaborative interaction. Hoccer[5] is an example of a simple collaborative scenario where users can share data by initiating a throw and catch gesture. The implementation of this scenario, which matches a throw and a catch fact, is shown by Listing 3. The rule declares that the throw and catch fact should originate from two different users (`nequal` test on line 4) and that the former should happen before the latter (temporal constraint on line 5). Finally, the spatial constraint on line 6 tests whether the throw was performed in the direction of the catch. Again, the recognition of multiple users concurrently throwing data at each other is completely handled by the inference engine without any additional programming effort.

Listing 3: Collaborative multimodal interaction

```
1  (defrule throwAndCatch
2       ?throw <- (Throw (user ?user1))
3       ?catch <- (Catch (user ?user2))
4       (test (nequal ?user1 ?user2))
5       (test (tSequence ?throw ?catch))
6       (test (sInDirectionOf ?throw ?catch))
7  =>
8       (assert (ThrowAndCatch
9            (user1 ?user1) (user2 ?user2)
10           (on:begin ?throw.on) (on:end ?catch.on)
11           (on ?catch.on))))
```

Compilation Rules are compiled to a Rete network to accommodate soft real-time performance. Rete is a very efficient mechanism that compiles multiple rules to a dataflow graph and stores intermediate results to speed up pattern finding problems. Note that the engine itself will take care of storing intermediate results, which usually puts an additional burden on application developers. It is also important to mention that the temporal and other constraints are handled at each node level. This means that we are open to incorporate more advanced approximation (e.g. based on data obtained by training) at runtime, without running into architectural or performance issues. This compilation step is provided by CLIPS and allows us to process an average of 9615 events per second with the two code samples (i.e. Listing 2 and Listing 3) active on an Intel Core i7 with 4GB of RAM. The data consisted of 80% Points, 10% Voice and 10% Throw/Catch facts with a successful fusion rate of 20%. The assumed data input in a realistic environment is around 25 (Point) +1 (Voice) +2 (Throw/Catch) events per second which implies that our engine definitely has no problems processing these scenarios in real-time. Figure 2 shows an example of the compiled Rete network for the rule defined in Listing 2. Note that the evaluation of depending matching is postponed whenever possible and the system only spends very little time for newly arriving event. However, this also implies that the ordering of the declared constraint in a rule can significantly influence the performance.

[5]Hoccer: `http://hoccer.com`

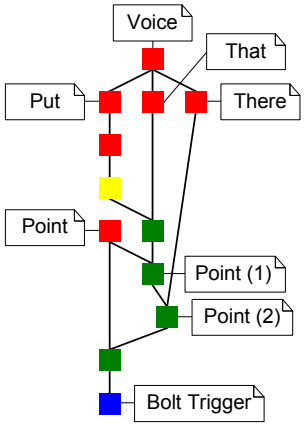

Figure 2: Compiled directed acyclic graph of Bolt's example

External Recognisers We also support the possibility to plug external recognisers into Mudra, along the fusion algorithms. Such external recognisers access data from the fact base and enrich it in turn. Recognition algorithms such as Dynamic Time Warping and Hidden Markov Models are examples of external recognisers. Flexible publish/subscribe handlers are provided for these recognisers. In future work, we would like to exploit this external recogniser feature even further via our concept of *smart activations*. As developers have access to low-level information in high-level fusion scenarios, it is easy to initiate additional low-level recognition techniques when desired. This initiation can embody computational intensive algorithms, which do not have to run continuously since their information is only interesting in certain scenarios. Typical applications are voice localisation using beam formation or the application of image processing for user identification.

3.3 Current Limitations

Our main focus is directed towards (1) finding meaningful patterns out of low- and high-level data, either via unimodal recognition or multimodal fusion, and (2) providing developers with high-level domain-specific language constructs to express advanced multimodal interaction with respect to the CARE properties [3]. Mudra supports a wide range of recognition techniques (DTW, HMM, production rules at the data-, feature- and decision-level), but it does not provide abstractions to set up a chain of raw data filters. We employ noise filtering for input data at the infrastructure layer (e.g. via a Kalman filter for accelerometer data), but we do not offer a complete infrastructure to chain stream boxes as for example offered by OpenInterface or Squidy. In case this is needed for future applications, it could be interesting to connect the output of these frameworks to our infrastructure layer.

A second limitation of our current implementation is the lack of advanced conflict resolution. A basic conflict resolution is offered via a numeric salience indication per rule, which allows developers to prioritise rules. However, since this construct only works when two rules trigger at the same time, we cannot always exploit this functionality. We also argue that a numeric salience value is insufficient to model all conflicting cases [15].

Dealing with probabilities at the attribute level permits a powerful control mechanism using high-level constructs. Although we

frequently exploit this explicit mechanism, we lack constructs to automatically reason over the combination of multiple probabilities, as implemented by fuzzy logic approaches. It is an open question to see whether the increase of complexity at the performance and programming level is worth the extra effort.

Developers using our unified multimodal framework have to be aware of the ordering of their conditional elements. As mentioned earlier, the order of conditions can significantly influence the performance. An initial, automated reordering of conditions is provided by CLIPS; however it is limited to trivial situations. To improve the performance, we propose a simple guideline: position events with a higher throughput at a later position in the rule and put conditions as close as possible to the matches, allowing the engine to avoid unnecessary event processing.

4. DISCUSSION

In this section, we discuss how Mudra's unified fusion relates to other existing approaches. Frameworks positioned at the data level, such as OpenInterface, Squidy and other data stream-based approaches, rely on the linear chaining of processing components. Although these *boxes* encapsulate the implementation complexity, the internal implementation of such a box is far from trivial. Suppose that a high throughput (vision) and a low throughput (speech) input stream have to be fused. The composition of such a box, which handles all events one by one, requires a lot of bookkeeping. Key events have to be kept in local variables (state management) and all combinations of matches have to be manually exploited (pattern matching). This ad hoc composition of boxes is of course feasible but puts a burden on the application developer who is only interested in expressing a simple correlation. This issue is particularly present when other developers would like to extend the internals of the box (e.g. to support multiple or collaborative users).

Most feature-level processing tools rely on preprocessed data, such as noise filtering or multi-touch identification, before the fusion occurs. However, this means that data-level processing can typically not benefit from recognised features as existing architectures enrol a one way propagation of events as illustrated in Figure 3. In Mudra, we benefit from a single fact base with a garbage collector, from which recognisers can access all available information at any time. Via this structure, data-level recognisers can incorporate optional feature data. It is worth mentioning that dealing with "optional" data is fairly easy to accomplish via rules. For instance, one rule is responsible for reasoning over raw data and another rule augments this data whenever additional features are found in the fact base. Due to the continuous evaluation of the inference engine, the second rule will automatically be triggered as soon as the feature information becomes available.

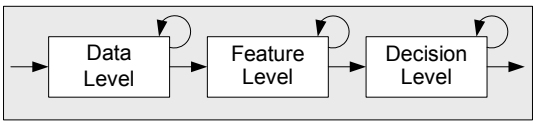

Figure 3: Traditional chaining of fusion

Decision-level multimodal fusion assumes the existence of high-level semantic data. This type of fusion is also known as *late fusion* where all high level information is gathered and correlated. Despite the introduced robustness, typical decision-level frameworks cannot recover from the loss of information which might occur at lower levels. A secondary limitation of these frameworks is the lack of support for overlapping matches. A commonly used implementation technique for high-level fusion is the incorporation of meaning frames. As already mentioned in Section 2.3, apparent issues arise when dealing with overlapping matches and continuous information. An important assumption of decision-level frameworks is the atomic compilation of meaningful events by data- and feature-level recognisers. However, for continuous gestures, such as pointing, it is hard to control the frequency. The continuous pointing in Bolt's "put that there" scenario is a simple example to stress this issue. The refresh rate of the pointing gesture is declared at the data-level processing. A high refresh rate introduces the problem of occupied slots in meaning frames, while a low refresh rate can lead to skipped decision-level integration since they are invalidated by the temporal constraints. It should be possible to circumvent these problems with an ad hoc solution, however, for more complex scenarios such as a multi-user environment, existing meaning-frame based solutions cannot be employed without inherent support for overlapping matches.

We argue that current frameworks are bound to their implementation approach, which means they can only offer well-defined abstractions either at the data-, feature- or decision-level. There is typically a one-way chaining from the lower to the higher level as shown in Figure 3. We have incorporated existing techniques in our unified approach and offer developers powerful language constructs to express their multimodal fusion requirements. One of the important benefits is that developers are freed from the manual bookkeeping of events. The declarative rules support the definition of multimodal fusion in terms of conditions on one or more primitive events via composite high-level rules. All recognisers build on top of each other, while they are still able to access low- or high-level information to improve their recognition rates. We also explicitly enforce every fact to be annotated with a timestamp for fine-grained garbage collection.

Our unified approach solves a number of important issues. However, there is still a lot of room for improvements and future research. For instance, we plan to evaluate the use of multiple recognisers on the same data. A combination of rules with multiple machine learning techniques that reason over the same data could significantly improve recognition rates.

Another issue that we are currently investigating is the incorporation of user feedback via supervised gesture learning. Since all low-level and high-level information is available in the fact base, rules could be used to manage user feedback intentions. Whenever such a rule is triggered, we could delegate the training process of gesture recognisers using knowledge of previous handling. Additionally, a batch learning approach can be used, in which the training of a gesture is only triggered with a threshold number of positive and negative examples.

Finally, we would like to include the smart activation of input modalities. For example, a low-level movement sensor could trigger the activation of a 3D camera, which in turn could activate the speech recognition module whenever a user is close to the microphone to improve speech recognition rates. The same *smart activation* constructs could also be used to control the propagation of information to machine learning techniques, as they are computationally too expensive for continuous evaluation.

5. CONCLUSION

Multimodal interfaces have become an important solution in the domain of post-WIMP interfaces. However, significant challenges still have to be overcome before multimodal interfaces can reveal their true potential. We addressed the challenge of managing multimodal input data coming from different levels of abstraction. Our investigation of related work shows that existing multimodal fusion approaches can be classified in two main categories: data

stream-oriented solutions and semantic inference-based solutions. We further highlighted that there is a gap between these two categories and most approaches trying to bridge this gap introduce some ad hoc solutions to overcome the limitations imposed by initial implementation choices. The fact that most multimodal interaction tools have to introduce these ad hoc solutions at one point confirms that there is a need for a unified software architecture with fundamental support for fusion across low-level data streams and high-level semantic inferences.

We presented Mudra, a unified multimodal interaction framework for the processing of low-level data streams as well as high-level semantic inferences. Our approach is centered around a fact base that is populated with multimodal input from various devices and recognisers. Different recognition and multimodal fusion algorithms can access the fact base and enrich it with their own interpretations. A declarative rule-based language is used to derive low-level as well as high-level interpretations of information stored in the fact base. By presenting a number of low-level and high-level input processing examples, we have demonstrated that Mudra bridges the gap between data stream-oriented and semantic inference-based approaches and represents a promising direction for future unified multimodal interaction processing frameworks.

Acknowledgements

The work of Lode Hoste is funded by an IWT doctoral scholarship. Bruno Dumas is supported by MobiCraNT, a project forming part of the Strategic Platforms programme by the Brussels Institute for Research and Innovation (Innoviris).

6. REFERENCES

[1] R. A. Bolt. "Put-That-There": Voice and Gesture at the Graphics Interface. In *Proc. of SIGGRAPH 1980, 7th Annual Conference on Computer Graphics and Interactive Techniques*, pages 262–270, Seattle, USA, 1980.

[2] J. Chai, P. Hong, and M. Zhou. A Probabilistic Approach to Reference Resolution in Multimodal User Interfaces. In *Proc. of IUI 2004, 9th International Conference on Intelligent User Interfaces*, pages 70–77, Funchal, Madeira, Portugal, 2004.

[3] J. Coutaz, L. Nigay, D. Salber, A. Blandford, J. May, and R. Young. Four Easy Pieces for Assessing the Usability of Multimodal Interaction: The CARE Properties. In *Proc. of Interact 1995, International Conference on Human-Computer Interaction*, pages 115–120, Lillehammer, Norway, June 1995.

[4] P. Dietz and D. Leigh. DiamondTouch: A Multi-User Touch Technology. In *Proc. of UIST 2001, 14th Annual ACM Symposium on User Interface Software and Technology*, pages 219–226, Orlando, USA, 2001.

[5] B. Dumas, D. Lalanne, and S. Oviatt. Multimodal Interfaces: A Survey of Principles, Models and Frameworks. *Human Machine Interaction: Research Results of the MMI Program*, pages 3–26, March 2009.

[6] F. Echtler, M. Huber, and G. Klinker. Hand Tracking for Enhanced Gesture Recognition on Interactive Multi-Touch Surfaces. Technical Report TUM-I0721, Technische Universität München, Department of Computer Science, November 2007.

[7] C. L. Forgy. Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem. *Artificial Intelligence*, 19(1):17–37, 1982.

[8] M. Johnston and S. Bangalore. Finite-State Methods for Multimodal Parsing and Integration. In *Proc. of ESSLLI 2001, 13th European Summer School in Logic, Language and Information*, Helsinki, Finland, August 2001.

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

[10] W. König, R. Rädle, and H. Reiterer. Squidy: A Zoomable Design Environment for Natural User Interfaces. In *Proc. of CHI 2009, ACM Conference on Human Factors in Computing Systems*, pages 4561–4566, Boston, USA, 2009.

[11] D. Lalanne, L. Nigay, P. Palanque, P. Robinson, J. Vanderdonckt, and J. Ladry. Fusion Engines for Multimodal Input: A Survey. In *Proc. of ICMI-MLMI 2009, International Conference on Multimodal Interfaces*, pages 153–160, Cambridge, Massachusetts, USA, September 2009.

[12] S. Oviatt. Advances in Robust Multimodal Interface Design. *IEEE Computer Graphics and Applications*, 23(5):62–68, September 2003.

[13] S. Oviatt. Multimodal Interfaces. In *The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications, Second Edition*, pages 286–304. Lawrence Erlbaum Associates, 2007.

[14] E. Petajan, B. Bischoff, D. Bodoff, and N. Brooke. An Improved Automatic Lipreading System to Enhance Speech Recognition. In *Proc. of CHI 1988, ACM Conference on Human Factors in Computing Systems*, pages 19–25, Washington, USA, June 1988.

[15] C. Scholliers, L. Hoste, B. Signer, and W. D. Meuter. Midas: A Declarative Multi-Touch Interaction Framework. In *Proc. of TEI 2011, 5th International Conference on Tangible, Embedded and Embodied Interaction*, pages 49–56, Funchal, Portugal, January 2011.

[16] M. Serrano, L. Nigay, J. Lawson, A. Ramsay, R. Murray-Smith, and S. Denef. The OpenInterface Framework: A Tool for Multimodal Interaction. In *Proc. of CHI 2008, ACM Conference on Human Factors in Computing Systems*, Florence, Italy, April 2008.

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

[18] T. Sowa, M. Fröhlich, and M. Latoschik. Temporal Symbolic Integration Applied to a Multimodal System Using Gestures and Speech. In *Proc. of GW 1999, International Gesture Workshop on Gesture-Based Communication in Human-Computer Interaction*, pages 291–302, Gif-sur-Yvette, France, March 1999.

[19] M. Vo and C. Wood. Building an Application Framework for Speech and Pen Input Integration in Multimodal Learning Interfaces. In *Proc. of ICASSP 1996, IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 3545–3548, Atlanta, USA, May 1996.

[20] G. Welch and G. Bishop. An Introduction to the Kalman Filter. Technical Report TR 95-041, Department of Computer Science, University of North Carolina at Chapel Hill, 2000.

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