The Nonverbal Toolkit: Towards a Framework for Automatic Integration of Nonverbal Communication into Virtual Environments

Frank Honold, Felix Schüssel, Kalina Panayotova and Michael Weber
Ulm University, Institute of Media Informatics, 89069 Ulm, Germany
Email: {frank.honold, felix.schuessel, kalina.panayotova, michael.weber}@uni-ulm.de

Abstract—To fully utilize the social potential of virtual environments, support for seamless and automatic integration of nonverbal communication is essential. In this paper we propose a conceptual and architectural design for a framework, which provides an integrated, flexible environment for the cooperation of heterogeneous modules specialized in different aspects of the acquisition, analysis and presentation of nonverbal communication cues. We propose a semantic framework for the knowledge exchange between these modules, as well as a general purpose architecture for the semantically correct fusion of nonverbal features.

Keywords—nonverbal communication; emotion; avatar; multimodal communication; social interaction; modeling

I. INTRODUCTION

Shared virtual environments are increasingly gaining attention and being recognized as a powerful tool for social interaction and on-line collaboration. Surprisingly, few improvements have been made in terms of possibilities for nonverbal communication (NVC). The lack of natural nonverbal behavior in avatars not only increases the ambiguity of verbal messages, it also often impedes the actual communication process in virtual environments, as the user is required to baby-sit their avatar by using low fidelity input interfaces.

To fully utilize the social potential of virtual environments, support for seamless and automatic integration of NVC is essential. In this paper we propose a conceptual and architectural design for a framework, which we call “The Nonverbal Toolkit”, serving as an integrated environment for the acquisition, semantic analysis and presentation of nonverbal communication cues in arbitrary virtual environments.

The basic principle of our design is strict separation of the extraction, processing (analysis, fusion, manipulation) and presentation of various NVC cues. This motivates the three main cornerstones of our design:

1) The framework should be able to utilize various information sources, such as multimedia capturing devices, external applications or even bio-signal measuring devices. The information made available by these various information sources should be analyzed and integrated in a meaningful way before it is presented to a target virtual environment.

2) The framework should be capable of providing the processed nonverbal information to various virtual environments – textual, visual, aural or mixed – in an adequate format, and should be open to integration of new target environments.

3) The framework’s design and data model should impose no or minimal restrictions in terms of NVC cues that can be extracted, processed and integrated into a virtual environment.

Since different virtual environments have different presentational capabilities, it is essential to explicitly describe nonverbal information and its semantics, and enable dynamic mapping of the information between processing components and the presentation elements of the particular target environment. A strong emphasis is also set on the possibilities for semantically correct fusion of information from multiple information sources.

II. RELATED WORK

Various approaches of how to extract and introduce nonverbal behavior into virtual environments are communicated in literature. Some approaches are based on linguistic analysis. Neviarouskaya et al. present an expressive 2D avatar for instant messaging, with the ability to recognize and to express emotions and to play social nonverbal behaviour on the basis of textual affect sensing and interpretation of communicative functions conveyed by online conversations [1], [2]. Cassel et al. [3] propose a Behavior Expression Animation Toolkit (BEAT) which is able to animate a human-like virtual body based on the linguistic and contextual information contained in the input text. Other approaches automate the nonverbal behavior of the avatar by treating it as a partially autonomous agent, reacting to various in-world conversation phenomena such as approach and initiation, requesting and giving feedback, planning etc. [4].

An interesting approach to facilitating (as opposed to automating) the integration of nonverbal behavior into virtual worlds is the Puppet UI interface, proposed in [5]. As an alternative input method a single-handed “puppet” is used to control the avatar’s nonverbal behavior. Other approaches for extraction of nonverbal information rely on body-tracking techniques, as proposed in [6] and [7]. However, both approaches do not explicitly model NVC cues and are therefore only applicable to avatar animation systems with support for avatar puppeteering.
Valuable insights about the modeling and integration of NVC can also be found in the research area of affective computing and intelligent conversational agents [8]–[10].

III. FRAMEWORK OVERVIEW

The Nonverbal Toolkit provides a flexible environment for the cooperation of heterogeneous expert modules specialized in different aspects of the acquisition, analysis and presentation of NVC cues. Functional modules are loosely coupled and easily recombined without loss of interoperability. Along with other plug-in components they are available for easy integration through a simple automatic or semi-automatic discovery process. Figure 1 illustrates the overall architecture and basic elements of the framework. The components are described following their numbering.

Fig. 1. Overview of The Nonverbal Toolkit

1) The framework manages a registry for the input sources, available for use within the framework. Input sources include input devices for audio and video as well as external applications, being able to provide valuable information about NVC cues.

2) The actual acquisition and analysis of the information, provided by input sources is performed by plug-in modules, which are managed in a repository and can be queried and deployed dynamically at runtime. Each module has a specific task and domain of expertise. Modules and the registry are described in section V.

3) Available modules can be combined in various ways to perform different processing tasks. Their orchestration and configuration is described in a simple configuration file.

4) Based on a particular configuration modules are wired-up to form a processing pipeline, described in section VI. Within the pipeline they are able to exchange knowledge in a well-defined knowledge representation format and with clear-cut semantics, as described in section IV.

5) Knowledge provided by different sources can be combined in a semantically meaningful way within a special module – the fusion engine, as described in section VII.

6) At the end of the processing pipeline the extracted nonverbal communication cues are translated and forwarded to target virtual environments by specialized modules. Examples for target virtual environments are avatar animation systems and virtual worlds such as Second Life, but also text-based environments such as social platforms, micro-blogs, speech generators or even simple export files.

IV. SEMANTIC FRAMEWORK

The cooperating modules need a common semantic model in order to exchange and combine their knowledge. This underlying model needs to support the sufficient expressiveness, modularity and extensibility, as well as the necessity for semantically solid reasoning in a domain characterized by uncertainty. The problem of defining such a model can be divided into three successive modeling stages: specification of a conceptual model, definition of a shared vocabulary and definition of application models. These three modeling stages are described in the remainder of this section along with a sample semantic model for the domain of human emotions which was created as an illustration of the semantic facilities of our framework.

A. Conceptual model

A conceptual model is a systematic description of the domain of interest, mainly intended to aid human understanding and communication. For example, several different conceptual models for the domain of human emotions exist. The categorical (nativist) model of human emotions defines a set of basic emotions and/or a hierarchical categorization whereas the model of affective dimensions views emotions as points in the multi-dimensional space spanned by two or more continuous axes of human affective states [11], [12].

As a conceptual model for the domain of human emotions, we have chosen the categorical approach where a number of universal basic emotions are identified and all other emotions are defined as sub-types or combinations of these basic emotions [11], [12]. The model also involves related aspects such as physical features, influenced by specific emotions, moods (longer lasting emotional tendencies), which are partially influenced by personality traits, as well as contextual features (see figure 2).
B. Shared vocabulary

A shared vocabulary is based on a conceptual model and provides a formal, machine interpretable description of domain concepts, entities and their interrelations, thus building a common semantic grounding for cooperating modules and enabling interoperability.

It is obvious, that the choice of one single shared vocabulary imposes the ultimate limitation on the framework’s capability to process various NVC cues, as well as on their possible level of detail. Therefore, instead of using one single, system-wide shared vocabulary (based on potentially different conceptual models) to be used, combined and referenced by cooperating modules. In addition to flexibility, this approach offers the advantage of allowing modules to specify their domain of expertise very precisely, so that only modules that are truly semantically compatible are wired-up within the processing pipeline.

Our research has revealed only a small number of existing knowledge representation models for the domain of NVC. Among the XML-based languages most are strongly oriented towards avatar animation systems, e.g. the Behavioral Markup Language [10], Virtual Human Markup Language [13], Affective Presentation Markup Language [14], and the Multimodal Presentation Markup Language [15]. Others, such as UserML [16] and EmotionML [17] lack the required expressiveness and maturity. Some examples for ontological modeling of concepts from the domain of NVC and emotions can be found in [18–21].

For the specification of shared vocabularies for the Nonverbal Toolkit the ontological approach was favored over a possible XML-based alternative. On the one hand, ontological languages offer a desired syntactic and structural independence. On the other hand, RDF vocabularies and OWL definitions have an incremental, distributed nature and offer suitable modularization and extension facilities. Knowledge from different information sources can be easily integrated into a global RDF semantic graph and then queried using an RDF query language such as SPARQL.

The Nonverbal Toolkit defines a top-level ontology describing the framework’s underlying data model. It encompasses the top-level class nvt:Feature1 – the common denominator for all types of observable or derivable aspects of NVC – and a set of basic attributes, as described in section VI. All domain specific ontologies derive their concepts from this class. This way existing ontologies can be easily integrated and reused for knowledge exchange in a modular way through simple subclassing and specialization.

Figure 3 shows an overview of our ontology for the domain of human emotions. All relevant classes are declared as subclasses of em:Feature. Note that this ontology not necessarily has to be aware of the top-level framework specific ontology mentioned above. It can be related to the framework’s data model through a simple mapping ontology declaring em:Feature as a subclass of nvt:Feature and specifying further restrictions on the attribute values, which the different feature types can assume.

![Fig. 3. High-level overview of the emotions ontology](image)

Inspecting the ontology into a deeper level in figure 4 reveals that basic emotions are specified as classes rather than individuals. Emotions assigned to different users or observed at a different time are thus regarded as separate instances of the same class. This design offers several advantages. Firstly, we are able to model the emotional state in a multi-sensor and/or multi-user environment without resorting to RDF reification or any similar constructs, which considerably affect representation and inference complexity. This is visible from listing 1 which represents an observation of a user’s emotion, modeled as an individual instance within an RDF knowledge package. This instance can then be easily referred to an existing sensor/module and user (lines 9 and 10 respectively).

```
1 <rdf:RDF>
2  <em:SadnessHappinessEmotion rdf:ID="sh423454">
3   <em:isInState>
4     <em:SadnessHappinessEmotion rdf:ID="sh423454">
5       <em:IntensityState>
6         <em:hasIntensityValue>5</em:hasIntensityValue>
7         <em:hasIntensityState>
8           <em:hasConfidence>0.9</em:hasConfidence>
9           <nvt:hasSource rdf:resource="http://fraunhofer.de/shore"/>
10          <nvt:hasUser rdf:resource="http://nvt.com/users/1"/>
11          <nvt:hasTimeStamp>2009-03-25T21:32:52.12679</nvt:hasTimeStamp>
12        </em:SadnessHappinessEmotion>  
13     </em:isInState>  
14   </em:SadnessHappinessEmotion>  
15 </rdf:RDF>
```

Listing 1. RDF knowledge package, observation of a bipolar emotion.

And secondly, such representation allows for further refining of the ontology by specifying specializations or combinations of basic emotion classes through inheritance mechanisms.

C. Application model

Application models provide the local, operational knowledge representation used by applications (here: modules, see sec. V) when processing information from their domain of interest. These models must be built on top of, or provide a mapping to a shared vocabulary. They are as detailed as the specific application requires. In the case of the Nonverbal Toolkit a partial Bayesian model for reasoning is derived and built from an ontology which is used as shared vocabulary, as shown in section VII.

V. MODULES AND REGISTRY

Each Nonverbal Toolkit module is an independent, encapsulated data processing unit operating within a limited semantic
subdomain. Due complying with the Nonverbal Toolkit’s module interface specification each module can register itself as a plug-in and can therefore be utilized within the toolkit’s processing pipeline. The common interface provides the means for a module to interact with the framework and to implement its life cycle events. Inter-module communication is defined via special input/output port interfaces. Implementations of these interfaces are provided by the framework and are assigned to involved modules during pipeline assembly.

To facilitate module implementation and to increase robustness, the framework effectively separates the actual information processing from the input/output and life cycle management by providing a special container module. It wraps around and controls other module cores – passive data processing components, provided by any module developer.

An essential part of a module’s specification is the meta information describing e.g. its capabilities and domain of expertise, system requirements as well as its compatibility with other modules. This information is provided as an RDF document and is used by the framework for module discovery. Using a predefined ontology, or even a restricted subset of it, a module is able to specify a set of shared and supported vocabularies as knowledge representation for its input and/or output. Such a subset can be defined through a SPARQL expression or through a list of supported features. Modules are also able to specify diverse system requirements, such as a specific operating system, input and output devices, external third party applications, and target environments etc.

Based on a module’s RDF description the Nonverbal Toolkit is able to determine its semantic support and thus validate a pipeline configuration for compatibility between modules. Furthermore, using logical inference based on the combined RDF model of the available modules, the framework’s configurator is able to suggest and construct intelligent pipeline configurations or optimize existing ones.

VI. DATAMODEL AND PIPELINE

Information processing within the Nonverbal Toolkit follows a simple pattern: knowledge is acquired by input modules and then sequentially combined, enriched and refined before it is presented through an output module. For this purpose cooperating modules are organized into a pipeline architecture where every module represents an atomic processing step and data is forwarded to the next processing step(s) via data channels, the pipes.

Modules are implemented as active processing components. This provides the advantage of effectively decoupling them at the runtime level, i.e. enabling them to communicate asynchronously and to process data concurrently and independently. Additionally to drawing performance benefits from the system’s parallel processing capabilities, this approach closely matches the event-oriented data model for NVC described below. Moreover, as a future extension of the framework, this architecture facilitates the realization of scalable distributed pipelines, where input and output modules can reside on different machines.

The framework’s data model is based on an event-oriented data flow. Even though in most cases input modules would analyze a periodic stream of raw data, the results of their analysis describe irregular patterns. This is due to the fact that the output reflects features extracted from the user and its social environment which changes do not occur in an well-defined frequency, but happen unpredictable and sporadic. While sampling rates for video and audio streams are based on human perception characteristics, there are no similar guidelines for defining an appropriate rate for the various nonverbal information features. With our event-based approach the system is capable to process the information just when an observed feature is extracted. Subsequent to extraction, features can then further be analyzed, transformed, aggregated, or correlated for complex pattern detection. Modules doing this work along the pipeline account for improvement of information quality or for deriving higher-level features.

The Nonverbal Toolkit’s data model is designed to address the issues resulting from this complex event processing problem. Events are modeled in a top-level ontology which forms the common base for all domain specific ontologies. It defines the basic class nvt:Event as well as its essential properties.

Each event within a knowledge exchange package is marked with a hasSource, hasConfidence, and a hasTimeStamp property. The data model provides two additional properties for the control of event validity. The property hasMaxDuration specifies the maximum time span for which the observation is considered as valid. After this period of time the confidence of the event should start to decrease. The property hasDelay indicates that the caught event must not be considered valid before the specified time has elapsed. For example, consider a module core specialized in recognizing simple hand gestures based on a sequence of hand positions. By the time a valid gesture is recognized, the
module core would be ready to generate two events at one time – one marking the beginning of the gesture and one marking its end. Obviously, the second event needs to be delayed in relation to the first in order to convey its proper meaning.

Even though this is not explicitly reflected in the ontology, logically one can distinguish between simple and complex events. A simple event is a primary, original event generated by an input module. Complex events are generated as a reaction to other events, e.g. as the result of a fusion process and implicitly reflect knowledge obtained through a set of simple events. The five event properties $\text{hasMinLatency}$ (latency of the constituent event with the lowest latency), $\text{hasMaxLatency}$ (latency of the constituent event with the highest latency), $\text{hasAvgLatency}$ (average latency of constituent events), $\text{hasLatencyVariance}$ (variance of constituent events’ latencies) and $\text{hasSimpleEventCount}$ (number of constituent simple events) characterize an event with respect to its current status. This information is recalculated after each processing step and can be used by modules to determine how an event should be handled. It can help consuming modules to determine whether different complex events refer to the same situation or not. In this case the same situation is seen as a set of real-world circumstances which triggers an event. In addition, the information can be used for more precise temporal alignment and as a factor for confidence calculation of derived events. The temporal correlation (or lack thereof) is considered as an information value itself and is further exploited as a part of the fusion process. The four latency parameters mentioned above represent an approximation of the latency distribution of the underlying fused simple events.

For simple events, the four distribution variables listed above are implicitly equivalent to the current event latency. For complex events, they can be easily calculated based on the distributions of the fused simple events. If $X = (x_1, x_2, \ldots, x_n)$ and $Y = (y_1, y_2, \ldots, y_n)$ build the set of simple events fused within two complex events $E_X$ and $E_Y$, and $Z = X \cup Y$ is the set of simple events fused within the derived event $E_Z$, the arithmetic average (mean) and variance of $E_Z$ are given through

$$\mu_Z = \frac{\mu_X |X| + \mu_Y |Y|}{|X| + |Y|}$$

$$\sigma_Z^2 = \frac{\sigma_X^2 + (\mu_Z - \mu_X)^2 |X| + \sigma_Y^2 + (\mu_Z - \mu_Y)^2 |Y|}{|X| + |Y|}$$

where $\mu$ denotes the true mean, $\sigma^2$ the variance and $|X|$ the cardinality of a data set. Figure 5 shows a graphical representation of the complex event latency analysis.

The event generator (the module core) can explicitly provide the latency distribution for derived events, thus ensuring that only relevant events are taken into account. If the information is not explicitly provided, the core wrapping module container will calculate it based on the most recent event input.

Even though processing modules “catch-up” on queued events through reading all accumulated data from their input pipe and performing temporal aggregation, at this point the Nonverbal Toolkit does not provide an explicit mechanism for meeting real-time constraints. Events are processed on a “best effort” principle and resulting delays are accumulated within the latency distribution information attached to each event. Based on this information and the semantics of the updated features output modules are able to estimate the timeliness of the events received and to decide how to react.

A. Adaptivity

To handle disconnected input or output modules as well as failed ones within the pipeline chain the framework adapts the process via an invalid path elimination. As a consequence even indirectly unavailable modules within the process graph get removed to ensure a proper dynamical adaption of the processing pipeline’s structure. Besides the ability of appropriately dealing with a particular situation (e.g. if a new input device or component becomes available) this mechanism also preserves the pipeline’s consistency under erroneous conditions (a component fails, lighting conditions deteriorate etc.).

VII. INFORMATION FUSION

With the exception of minimalistic and trivial use cases, data often has to be rated and combined to infer further information aiming to achieve an accurate representation of an observed situation. For this purpose we introduce a special type of input-output module – the fusion module – responsible for integration and fusion of information, conveyed from different competitive and complementary sources.

Since from the system’s point of view the fusion engine is a regular input-output module, diverse techniques and implementations can be used to solve the various fusion challenges. The method we propose in the remainder of this section represents a general purpose approach towards semantic fusion on the feature level.

A. Overview

A fusion module is a special type of input-output module which can be regarded as a semantic synchronization point within the processing pipeline. A fusion instance can be...
inserted anywhere in the process pipeline, where information from different sources pertaining to the same semantic domain needs to be combined and enhanced. Figure 6 illustrates the main building blocks of an exemplary fusion module and the interaction during a cycle of the fusion process. The enumeration below refers to the fusion process’s stages in figure 6. The process works as follows:

1) At this stage incoming data is transformed to an internal RDF representation for further processing. Typically this involves constructing an RDF graph from a serialized RDF syntax such as Notation3, Turtle or RDF/XML. During this process, a validation against the fusion module’s declared ontology is performed. The RDF graph fragment is then passed forward to one or more so-called update vectors.

2) Each update vector provides a slot for every variable of the application model (here a Bayesian network, stage 3), which can be updated by a module. More specifically, all observations about a single model variable received by a single input module are directed towards the same slot. If necessary, observations are integrated over a specific time interval, before an update of the actual Bayesian model is triggered. A special set of slots is provided for the computation of the so called history variables (depicted in light gray in figure 6). These variables are used to track the evolution of selected features over time and are included in the Bayesian network model to provide for a certain level of stability specific to each feature.

3) Preprocessed information contained in the update vector (stage 2) enters the application model of the fusion module, where the actual feature fusion takes place. For this purpose a Bayesian network model is used. Based on available evidence the Bayesian network computes the probabilistic estimates of all model variables. After each update step model changes are introduced to an RDF representation of the model at stage 5. Furthermore, the estimated value for each variable tracked over time is fed into its corresponding slot within the update vector, thus enabling the computation of its history value in the next iteration.

4) Maybe the application model could be broken down into several independent or hierarchically arranged sub-models. In this case, the stages 2 and 3 are repeated for every sub-model. In addition to potentially reducing computation cost this approach facilitates both: model design and learning through explicitly dividing the model into manageable pieces.

5) At stage 5 the inferred up-to-date knowledge is represented through a common vocabulary, namely an RDF graph complying with the fusion module’s declared ontology. The RDF model provides a subscription mechanism for observers, interested in specific model changes (stage 6).

6) At the final stage 6 model observers are notified about the model’s changed state. The observer is then able to query the model for variables relevant to its corresponding module, adapt and serialize the information if necessary, and pass it along the pipeline.

B. Bayesian representation of a feature

Since information about the same feature can be provided by multiple sources, the model is required to handle competing and, possibly, contradicting or ambiguous information. Therefore, instead of resolving the conflict externally through an explicitly computing a compromise value, the application model integrates the information sources into the Bayesian network. The feature itself is a latent variable, which is to be estimated based on a given set of observations. A general representation of a feature within the Bayesian model is shown in figure 7.

Each observation variable belongs to only one parent, namely the feature variable being observed, and it duplicates the set of possible states of this parent. Thus, given the parent feature, the observation is conditionally independent of any other network variable. In this way the only information en-
A general difficulty arises from the integration of features which need to be tracked over time. A typical Dynamic Bayesian Network (DBN) operates with a fixed evolution rate, since different nonverbal features evolve with very different rates. We recommend to address this issue by introducing a set of history variables instead of explicitly representing each state of a feature’s temporal evolution within the model, i.e. constructing a DBN. In our approach information from a specific time interval is condensed in a single approximation variable.

VIII. PROTOTYPICAL IMPLEMENTATION

As part of our research, we have developed an early horizontal prototype of the Nonverbal Toolkit. Its main purpose is to provide a working environment for the realization of end-to-end processing pipelines enabling incremental experiments and validation of the framework’s design and data model.

The prototype’s implementation is based on the OSGi Alliance² standard, which closely matches the requirements for the system’s modularity and extensibility.

The minimalistic knowledge model used for our experiments did not require the expressiveness of OWL/RDF modeling, therefore to reduce the prototype’s complexity XML was chosen as the format for knowledge exchange at this early stage. The prototype provides a simplistic graphical user interface, which can be used to load a configuration, and to control the pipeline’s execution.

A. Input

For the extraction of nonverbal cues the Sophisticated Highspeed Object Recognition Engine (SHORE) [22] was used. SHORE is a highly optimized software library for face and object detection developed at the Fraunhofer Institute for Integrated Circuits IIS. Within our framework this library is utilized for real-time extraction and classification of facial features and expressions from a video stream or from a still image. The SHORE library is integrated into a lightweight stand-alone application, directly connected to a web camera. The application’s output is made available to the respective Nonverbal Toolkit module via a socket connection. Additionally, a textual interactive input module was created as a simulation of a second input source.

B. Output

Currently, the Nonverbal Toolkit’s prototype is able to integrate within two target environments – OpenSimulator and the XfacePlayer. OpenSimulator [23], also known as OpenSim, is an open source 3D application server, which can be used to create a virtual environment, remotely accessible through a client application. OpenSim can be used to simulate a virtual environment similar to Second Life³ and since it supports its communication protocol it can be accessed through the standard Second Life client.

The OpenSim output module was realized in collaboration with IBM Deutschland R&D and Fraunhofer IuK as so-called GridProxy application. GridProxy [24] is a library which allows applications to wedge themselves between the official Second Life client and servers. GridProxy applications can inspect and modify any packet as it passes between the client and the servers, remove packets from the stream, and inject new packets into the stream. Hence, changes in the state of knowledge about a user’s nonverbal signals or emotional state can be transformed into appropriate animation packages and sent for execution to the OpenSim server through packet injection. The animations, currently supported by the module’s GridProxy implementation, correspond to four basic emotions.

The second experimental target environment used with the prototype is Xface. Xface is a set of open source tools for creation of MPEG-4 and keyframe based 3D talking heads [25]. The XfacePlayer can be controlled through SMIL-Agent scripts, locally or remotely through a socket connection. Xface was used to experiment with better control over the user’s facial expressions (see figure 8).

C. Fusion

As part of the prototype’s implementation and as a proof of concept a simple fusion engine with basic support for Bayesian inference was developed. In the core of the fusion engine’s implementation the Bayesian network software library NETICA by Norsys Software Corp. [26] was used. A simple Bayesian model was trained for the fusion of several features based on the Nonverbal Toolkit’s emotion ontology.

IX. CONCLUSIONS AND FUTURE WORK

We addressed the issue of enabling the integration of nonverbal communication signals into virtual environments in a seamless and flexible fashion. We discussed various aspects and challenges of modeling and combining nonverbal information from multiple sources and proposed a novel approach

---

²OSGi Alliance: http://www.osgi.org/Main/HomePage
³http://www.secondlife.com/~3D virtual world community
towards integration of nonverbal cues into various target virtual environments.

The construction of adequate semantic models and knowledge bases for the domain of nonverbal communication is a crucial issue, which is not yet resolved entirely. Another remaining research question is whether a global semantic model would provide better results than independent local models, which possibly require ontological mapping.

Furthermore, the framework’s data model is still to be fully validated for its plausibility regarding synchronization of nonverbal information, as our early experiments do not provide enough resources to do so.

Regarding the approach in general, a few limitations should be considered. Although communicative nonverbal behavior adheres to some general principles, it is far from being fully understood. Any computational models are therefore going to be relatively simplistic and constrain available behavior to a limited set of displays lacking many real world nuances. This raises concerns about the system’s capability to accurately reflect the user’s intentions under unforeseen circumstances or resolve issues of ambiguity.

This framework has the potential for diverse fields of application. Automatically integrated nonverbal behavior could greatly facilitate on-line collaboration in a shared visual environment such as virtual conferencing. An interesting future development of the framework would be to provide support for shared nonverbal and emotional user models, which would enable semantic fusion on a group scale, rather than on individual scale.

ACKNOWLEDGEMENTS

This work is originated in the Transregional Collaborative Research Centre SFB/TRR 62 “Companion-Technology for Cognitive Technical Systems” funded by the German Research Foundation (DFG) in collaboration with IBM Deutschland R&D, and Fraunhofer IuK.

REFERENCES


Author pre-print version. Please check the original proceedings for the final version.

IEEE proceedings of the 8th International Conference on Intelligent Environments (IE), Guanajuato, Mexico, June 2012