Multimed Tools Appl
DOI 10.1007/s11042-010-0530-2

Multimodal behavior realization for embodied conversational agents

Aleksandra Čereković · Igor S. Pandžić

Abstract Applications with intelligent conversational virtual humans, called Embodied Conversational Agents (ECAs), seek to bring human-like abilities into machines and establish natural human-computer interaction. In this paper we discuss realization of ECA multimodal behaviors which include speech and nonverbal behaviors. We devise RealActor, an open-source, multi-platform animation system for real-time multimodal behavior realization for ECAs. The system employs a novel solution for synchronizing gestures and speech using neural networks. It also employs an adaptive face animation model based on Facial Action Coding System (FACS) to synthesize face expressions. Our aim is to provide a generic animation system which can help researchers create believable and expressive ECAs.

Keywords Multimodal behavior realization · Virtual characters · Character animation system

1 Introduction

The means by which humans can interact with computers is rapidly improving. From simple graphical interfaces Human-Computer interaction (HCI) has expanded to include different technical devices, multimodal interaction, social computing and accessibility for impaired people. Among solutions which aim to establish natural human-computer interaction the subjects of considerable research are Embodied Conversational Agents (ECAs). Embodied Conversation Agents are graphically embodied virtual characters that can engage in meaningful conversation with human users [5]. Their positive impacts in HCI have been proven in various studies [16] and thus they have become an essential element of

A. Čereković (✉) · I. S. Pandžić
Faculty of Electrical Engineering and Computing, University of Zagreb, Zagreb, Croatia
e-mail: aleksandra.cerekovic@fer.hr

I. S. Pandžić
e-mail: igor.pandzic@fer.hr

computer games [18], intelligent educational systems [47] and new media such as Interactive Storytelling [46].

While conversing with humans, ECAs show multimodal nature of human-human conversation which includes both verbal and nonverbal channels. Thus, in order to have believable ECAs it is necessary to model both types of behaviors [32, 42, 49]. This has posed significant challenge before researchers, as nonverbal behaviors can be exceedingly complex and depend on a multitude of diverse factors, such as gender, personality, emotional state, social relations and cultural background.

Our research focuses on one of the crucial components which makes ECAs believable and expressive in interaction with humans [54], and that is the animation system which realizes multimodal behaviors. Our goal is to devise a realization system which satisfies the following requirements:

- **Real-time multimodal behavior execution**—realization of complex communicative utterances, which include verbal and nonverbal behaviors, in real time.
- **Animation Control**—intuitive high-level mechanisms for character motion control
- **Extensibility** with large number of animations. The feature is especially important for culturally codified ECAs.
- **Applicability**—applicable to a wide range of domains, such as ECAs involved in marketing, virtual tutors, advisors, as well as communicative non-player characters (NPCs) in role-playing games
- **Integrability**—multi-platform, modular and easy to integrate with existing engines and application frameworks
- **Availability**—open-source and publicly available for the benefit of the research community

Rather than devise a new language for describing physical realizations of multimodal human behaviors, we base our system on the existing Behavior Markup Language (BML) [3, 30, 53]. BML is developed as part of the SAIBA (Situation, Agent, Intention, Behavior, Animation) framework, an open framework targeted at researchers in the area of ECAs and designed in collaboration with foremost experts in the field of ECAs and human communication, including psychologists, sociologists and linguists.

BML defines multimodal behaviors in human-readable XML markup, where BML elements represent various primitive actions (e.g. speech, facial and body gestures). BML allows modeling of multimodal behaviors by specifying temporal relationships between these elements.

The use of BML affords us several distinct advantages:

– BML is intuitive, yet expressive enough to model complex behaviors
– it is fairly simple to implement
– it appears to have been well-received by the research community and is already being used by several research groups [53]

In this paper we discuss issues of multimodal behavior realization for ECAs with focus on motion control using BML. Based upon results of our previous work [8] we develop RealActor, the real-time behavior realization system for ECAs. Main contributions of our paper are the following: the full-body and open-source real-time multimodal behavior realization system, a novel solution for aligning animation with synthesized speech using neural networks, and adaptive face animation model based on Ekman's Facial Action Coding System (FACS) [14].

2 Related work

Theoretical background from linguistics, psychology, and anthropology has proven the great importance of nonverbal behaviors in face-to-face communication and their strong relationship with speech. For example, research findings show that eyebrow movements can emphasize words or may occur during word-searching pauses [13], iconic gestures accompany certain lexical units [34], co-expressive gesture and speech are shaped by dialectic between linguistic expressions and spatio-motoric representations [34] and synthetic displays are the most frequent facial gestures accompanying speech [9]. Therefore, we can conclude that in multimodal interaction humans use their speech and nonverbal signals whose acoustic features, spatial characteristics and the mutual time relations form the meaning of a multimodal behavior. In HCI research with regard to synchronization of single modalities, a lot of attention has been paid to data stream synchronization in real-time multimodal fusion. Example is a prominent work by Johnston et al. [27], who build an architecture which integrates spoken and gestural input using unification of features which represent semantic contributions of the different modes. In their architecture, synchronization of speech and gestures is done using time-marks and a finding that speech typically follows gesture within a window of three to four seconds while gesture following speech is very uncommon [38]. In another work from Johnston and Bangalore [26] a weighted finite-state device takes speech and gesture streams as inputs and then outputs their joint interpretation. The speech stream is processed after the gesture stream, since the majority of gestures may happen to be unambiguous deictic pointing gestures.

The synchronization process in realizing multimodal behaviors for Embodied Conversational Agents, in general, depends on description of behaviors through multimodal representation languages and the temporal references of behaviors of single modalities. For example, most of the existing multimodal behavior character animation systems incorporate nonverbal behavior generation rules from psychology using statistical models which automatically generate multimodal expressions from input text [6, 32, 36, 45] or speech [1, 57]. These systems work as black boxes—they use their internal representations of multimodal behaviors and produce speech and animations. When processing input text or speech, they usually use time stamps which tell the realization system when to start a specific animation in addition to the speech. The generation rules decrease the effort of ECA development, but provide no freedom for animators. After nonverbal behavior animations have been selected, if unsatisfying, they cannot be replaced with others. Moreover, animations cannot be isolated and used for another purpose. The only exception is the modular Nonverbal Behavior Generator [32] which has a layered architecture and uses the SmartBody module for behavior synthesis.

In multimodal behavior representation, as paper [30] evidences, over the last 10 years several descriptive multimodal representation languages have been formed. These languages have certain disadvantages and are mostly utilized only by their developers. After the formation of Behavior Markup Language (BML), which is an improvement and extension of these languages [30], the researchers have started to build BML compliant components which can be used by a number of ECA systems and exchanged inside the community. The main challenges that BML posed before developers are synchrony issues between modalities which may have complex correlations. So far three BML-compliant animation engines have been developed—ACE, SmartBody [51] and EMBR system [22]. Articulated Communicator Engine (ACE) developed from an earlier engine based on multimodal utterance representation markup language (MURML) [29]. It provides support for facial expressions, lip synchronization, visual text-to-speech synthesis and hand gestures. It can be integrated with different graphics engines and has been utilized in two different projects—NUMACK and Max. SmartBody supports facial expressions, speech, body postures, body gestures, gaze, head motions, feedback notifications and interruptions of running behaviors. These elements are implemented as hierarchical motion controllers that act on different body parts. The controller hierarchy also includes special meta-controllers for motion blending, time-warping and scheduling. SmartBody has been utilized in several projects and integrated with a variety of engines, including OGRE, GameBryo, Unreal 2.5 and Source. SmartBody is engine-independent and interacts with the rendering engine via TCP. The most recent of the BML-compliant realizers, EMBR system, is architecturally similar to SmartBody. It is based on a modular architecture that combines skeletal animation, morph targets and shaders. Novel solution to this work is EMBRScript control language, a wrapper between the BML language and realizer. The language uses time constrains to specify time and spatial constraints of the resulting animation. Animations are produced on the fly, using motions segments, which are blocks of animations computed from recorded animations, key frames, morph targets and the inverse kinematics system. EMBR relies on free Panda3D graphics engine and Blender modeling tool.

RealActor's architecture is designed to overcome the issue of mechanical-looking motions for ECAs displayed by most of existing systems [54]. It enables usage of example-based animations modeled by an artist or saved using motion capture data, which gives compelling results. RealActor's architecture is similar to SmartBody, but with several key differences. First, we introduce a new method of coordinating nonverbal behaviors with speech that employs machine learning techniques to achieve synchrony. Secondly, our system relies on a lexicon of animations which contain metadata necessary for mapping words to appropriate gestures during behavior planning. Our speech synthesis is based on Microsoft Speech API programming interface [35], by far the most popular text-to-speech system that is distributed as part of Microsoft Windows operating system and supported by virtually all commercial TTS engines. In effect, this means that our system is compatible with almost all commercial speech synthesis products in existence. And finally, our system employs an adaptive face animation model based on Ekman's FACS.

3 Multimodal behavior realization issues

In this section we introduce issues of building the multimodal behavior realization system with a focus on BML. Solutions which our system uses are further explained in Section 4 and Section 5.

3.1 Multimodal behavior specification with BML

BML defines a multimodal behavior as a block of interrelated BML elements. BML elements correspond to primitive behaviors, e.g. gesture, head, face, speech, gaze, locomotion etc. Synchronization of these different elements is possible because each element has six phases delimited with seven synchronization points—start, ready, stroke-start, stroke, stroke-end, relax, end. Synchronization relationships are specified using references to these points.

Let us consider the following BML script:
Example of a BML script

Example of a BML script

```
  <?BML version="1.0"?>
  <act>
  <bml id="bml2">
       <gesture id="g1" stroke="s1:tm" type="beat"
       hand="right" handlocation="outward;center;near"/>
       <speech id="s1">
            <text>  Trust  me,  I  know  what  I'm  <sync
id="tm"/> doing. </text>
       </speech>
       <head id="h1" stroke="s1:tm" action="rotation"
            rotation="nod" amount="1"/>
  </bml>
  </act>
```

In this example, the speaker utters the sentence “Trust me, I know what I’m doing.” (specified in the speech element). As he pronounces the word “doing”, the speaker should make a hand gesture (gesture element) and nod his head reassuringly (head element). More specifically, stroke points of the hand gesture and nod need to be aligned with the word “doing”.

Each BML element has attributes and their values define physical features of primitive behaviors. In this example, speaker will use his right hand to make the beat gesture. As can be seen, BML specifies no absolute duration of primitive units. In our system, they are defined in the animation database.

3.2 Key challenges and design guidelines

Key challenges posed by implementation include how to model synchronization across modalities (BML elements) and how to take into account events from the environment when realizing behaviors. There is also the issue of realizing two or more BML elements of the same kind simultaneously and synchronizing animation with speech synthesized using text-to-speech (TTS) system which does not provide a priori timings. The system’s design should also preserve modularity so the system can be easily extended with novel animation techniques or animation examples. Therefore we employ:

- **Two-stage processing for defining timings of BML elements.** Multimodal behavior realization process with BML scripts requires fine tuning of temporal relations between BML primitive units (behaviors), which are realized through animations of nonverbal behaviors and/or speech. Since primitive behaviors may have relatively complex relations, we believe that a pre-processing step, in which time boundaries for behaviors are defined, is necessary for effective multimodal behavior realization. Our two-stage processing includes parsing and preparing the elements for execution. RealActor parses BML code and then for each BML element, except synthesized speech, looks up corresponding animation in the Animation Database. During preparation, BML element cross-relations and animation timings are used to define the beginning of execution for each BML element.
- **Layered Controllers for synchronization across modalities.** RealActor is based on controller units responsible for realization of planned animations. The main controller

Behavior Scheduler uses timing information and time control to decide which behaviors will execute and when. By continually updating the execution phase of each behavior it is able to synchronize BML elements and provide feedback about progress of behavior execution. Underneath the Scheduler, there are several low-level controllers responsible for realization of primitive behaviors.

- **External Functions for behavior control.** In addition to the running behaviors the animation system should also respond to world events, especially those of the participating interlocutors. For that purpose we implement functions which control running behaviors. The behaviors can be stopped while running or they can be merged with new-coming behaviors, which are reactions to the environment events. Furthermore, RealActor supports BML's message feedback, so an external component, like BML behavior planner, can track the realization progress.
- **Machine learning techniques for coordination of synthesized speech and animation.** In the BML example from the previous sub-section the speaker pronounces the word "doing", makes a hand gesture (gesture element) and nods his head. More specifically, stroke points of the hand gesture and nod need to be aligned with the word "doing", as illustrated in Fig. 1. In other words, animations that correspond to these gestures must start playing in advance, so that the onset of their stroke phases corresponds with the uttering of the word "doing". To achieve this, two approaches can be applied. *Dynamic adaptation* attempts to synthesize necessary synchronization data on the fly [29], whereas *animation planning* requires a priori timing information that needs to be prepared before behavior realization [6, 20, 51, 52]. The latter approach is simpler to implement, but requires preprocessing and manual annotation of animation clips. We employ it in our system, relying on a database of annotated motion. However, that does not solve the problem of synchronizing nonverbal behaviors with synthesized speech, as most TTS modules do not provide a priori timing information for words and phonemes. Only a few TTS engines are capable of providing this data, such as Mary TTS [44] and Festival [50]. Microsoft SAPI, which we use in RealActor, only provides viseme, word and sentence boundaries in real-time. Furthermore, the planning approach cannot handle self-interrupting behaviors such as gaze and head motion, nor their synchronization with other behaviors. To address these issues, we use neural networks to determine timing data in real-time where it is not available a priori.

4 System overview

RealActor is implemented as part of visage|SDK character animation framework [40, 41] and has the following features:

– specification of character behaviors using BML scripts
– start, stop, schedule or merge behaviors via high-level, BML-compliant API

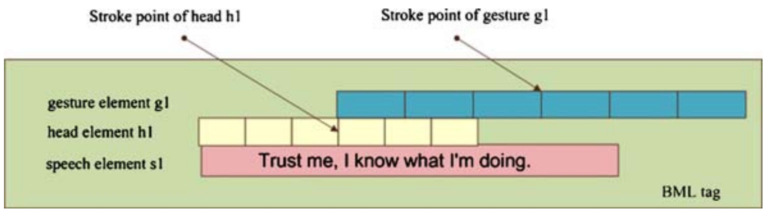

Fig. 1 Example of multimodal synchronization in BML

- database of annotated animations shared between multiple characters
- visual text-to-speech synthesis based on industry-standard Microsoft SAPI [35]
- lip synchronization
- supports animation using FACS
- supports integration with any engine or application via a minimal scene wrapper
- flexible art pipeline that includes tools for automated face model production from photographs [17] as well as automated facial morph generation and parametrization for models originating from commercial applications such as FaceGen [15]

RealActor has a layered architecture (Fig. 2). At the top of the system sits the BML realizer, tasked with parsing BML scripts, planning, scheduling and executing behaviors. Its functionality is described in detail in the next section. Below it is the motion playback system. Based on MPEG-4 FBA [39], it plays and blends animations that correspond to various elementary behaviors. Finally, the bottom layer is the scene wrapper, which provides a common minimal interface for accessing and updating the underlying character model. This interface needs to be implemented for different rendering engines. At the

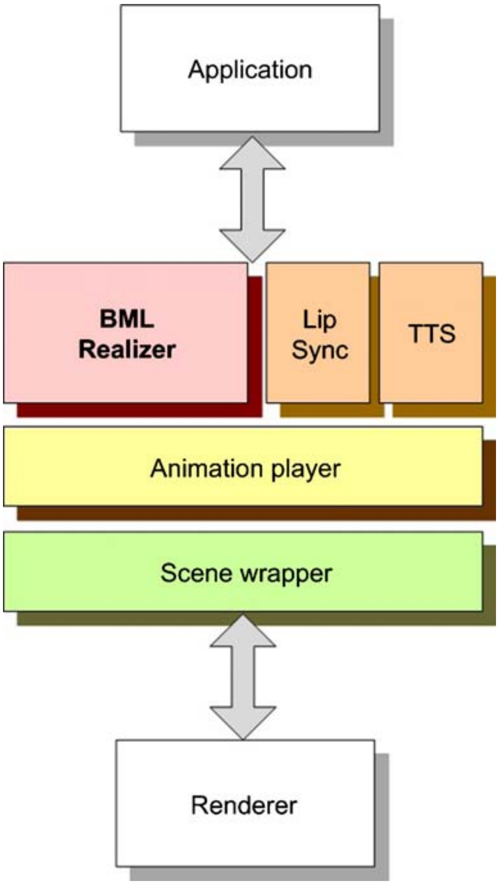

Fig. 2 RealActor system architecture - simplified view

moment, RealActor is supported in Horde3d [23], Ogre [37] and Irrlicht [25] rendering engines.

4.1 Multimodal behavior realization process

RealActor is carefully designed to drive realistic responsive and interruptive behaviors for ECAs (Fig. 3). The overall realization process is done by three components responsible for:

- parsing of BML scripts
- behavior preparation and planning
- behavior execution

First component*, BML Parser,* is a component responsible for reading BML scripts and creating objects—BML blocks containing BML elements with element attributes. BML blocks are added to a list which is then passed to Behavior Planner.

Behavior Planner does animation planning in two stages, as described in the previous section. It reads the list of BML blocks and prepares each block for execution by adding timing information needed for behavior synchronization. The planning is done using a recursive algorithm in which BML elements are processed using relative references to other BML elements and absolute timing information of BML elements which have been prepared for execution. This timing information is retrieved from the *Animation Database*, an animation lexicon which contains a list of all existing animations. In the Database each primitive animation is annotated with time constraints, type information and information about motion synthesis process (example/procedural). If input BML script contains a walking action, Behavior Planner uses *Locomotion Planner,* a subcomponent which handles character locomotion. It uses the character's current position and specified destination to compute the duration of the walking action.

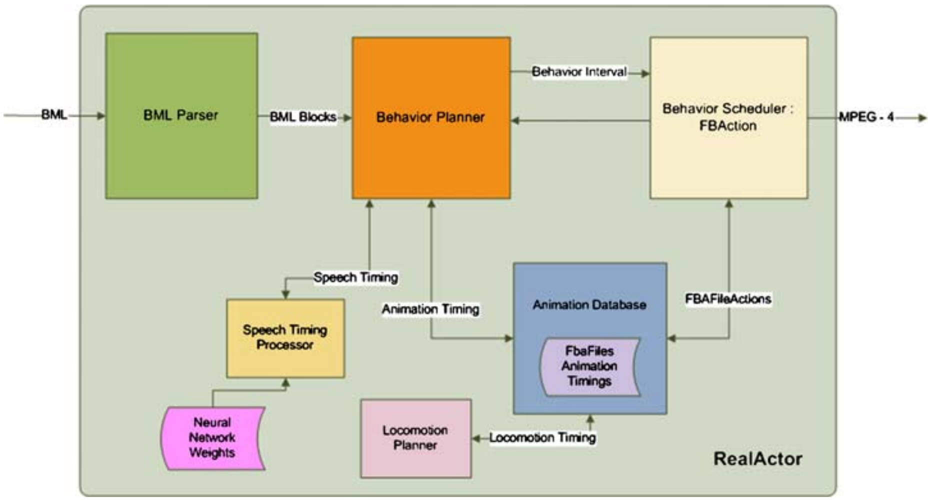

Fig. 3 Architecture of RealActor

The character's speech can be handled in two ways: using lip-sync or text-to-speech synthesis. If lip-sync is used, speech must be recorded in a pre-processing step and manually annotated with necessary timing information, which is then used for animation planning. Text-to-speech synthesis is more appropriate, because it can be performed concurrently with behavior realization. However, most TTS systems (including Microsoft SAPI used in RealActor) do not provide a priori phoneme and word timing information necessary for synchronization with nonverbal behavior. To address this issue we employ machine learning and neural networks, as described in the next subsection. The designed solution is implemented inside *Speech Timing Processor* component.

Finally, prepared BML elements are executed with TTS or lip-sync based speech and related animations (which are nonverbal behaviors) using a hierarchical architecture of animation controllers. The main controller, which we named *Behavior Scheduler*, uses timing information and time control to decide which behaviors will execute and when. By continually updating the execution phase of each behavior it is able to synchronize BML elements and provide feedback about progress of behavior execution. During realization, each BML block is realized sequentially. When one block is finished, the Scheduler notifies the Planner so a new BML block can be passed for execution.

The principle of realizing multimodal behaviors is based on applying the BML's primitive units to distinct body controllers, such as head, gesture, face... Behavior synchronization is handled by the Scheduler, which transmits motions to its low-level subcomponents responsible for realization of primitive behaviors (Fig. 4).

– *Gazing and head movements* are generated by two controllers, Gaze Controller and Head Controller, which run in parallel with the Scheduler. When new gaze direction or head movement occurs, the Scheduler sends to those two controllers parameter values which correspond to the new movement. Animation is executed depending on the current position of the character's head/eyes, target position and duration. The reason we have designed these control units is the nature of the movement of the head and eyes. During conversation, a person can sometimes make gestures that continue after his utterance is finished (meaning the body is not returned to the neural position). An example is gazing in multiuser communication; person A stops speaking and looks at person B. B then continues the conversation, which is then interrupted by person C. Person A, who was previously looking at person B, gazes at person C. This would mean that person A continues to look at person B after speaker's utterance is finished. At a time when C begins to speak, a new gaze action occurs. This event directly affects the synthesis of new movements that happen on the same modalities (eyes, head, body) because in RealActor, both head and gaze motions are designed as procedural animations which are generated on the fly using interpolation steps between the initial and target position. The implementation of head animations is partly taken from prior work [4]. We use head swings, nods and shakes And additionally, we have designed head tilts.
– To generate *facial expressions* we use two distinct controllers. The first controller, Face Controller, is compliant with the BML element face which controls mouth, eyes and eyebrows separately. Alternatively, we design a face animation model based on Ekman's Facial Action Coding System [14]. The model is controlled by AU Manager, which runs in parallel with the Scheduler, accepting and running Action Units (AUs). We have chosen Ekman's system because it is widely accepted in psychology and computer science andcan be used to synthesize an essentially infinite number of facial expressions.

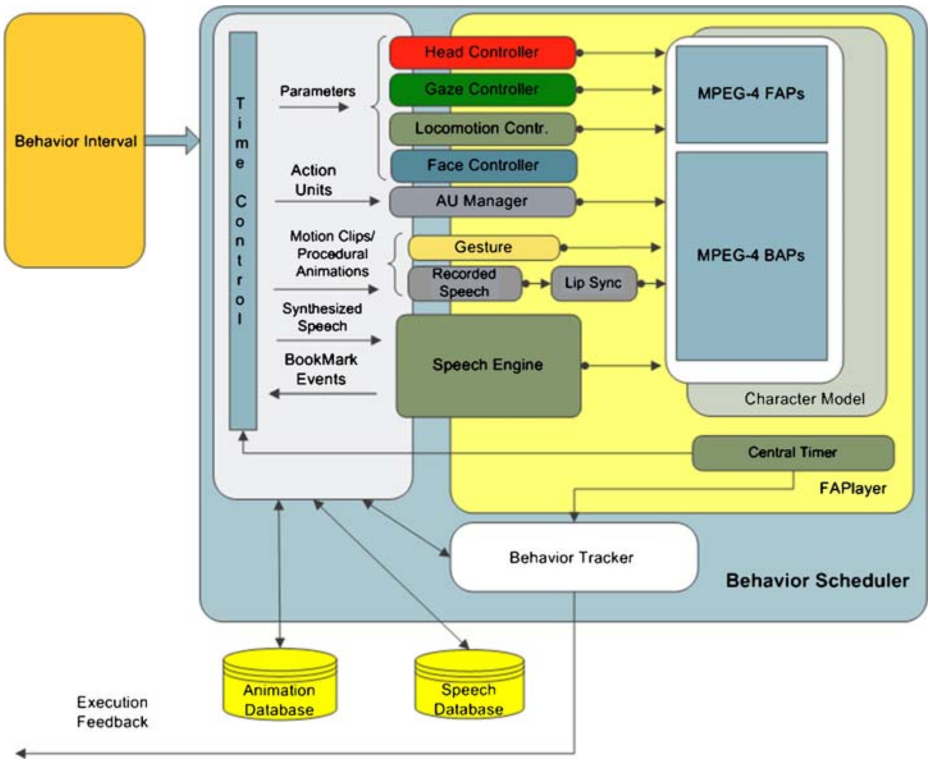

Fig. 4 Structure of Behavior Scheduler and its low-level subcomponents responsible for realization of primitive behaviors

- *Locomotion* Controller accepts target location and generates a walking action.
- Most of the *gestures* are generated using motion clips from the animation database containing 52 types of gestures [8]: beats, lexemes, conduit and generic gestures, which are complex symbolic animations controlled through external namespace elements in a BML script. The clips affect not only hands and arm movements, but also head movements which are blended with the result coming from the Head Controller.
- *Speech* can be generated either by applying our lip sync system [56] to audio clips from the Speech database or using text-to-speech synthesis engine. If developers want to use audio clips, they need to manually annotate timings of specific words that need to be aligned with gestures and store timing files in the animation database.

The body controllers are placed in *FAPlayer*, the character animation player in visage| SDK character animation framework [40, 41]. The player is responsible for blending streams of animation data in a meaningful way in order to generate responsive and believable behaviors. The player also contains a time processing unit which is essential to track behavior realization progress necessary to notify the Behavior Scheduler that it should start or stop execution of primitive units.

4.2 Playing and controlling animations

RealActor animation playback system is based on MPEG-4 Face and Body Animation standard [39], which specifies skeleton and face movements separately. Skeleton movement is parameterized with body animation parameters (BAPs), each controlling a specific degree of freedom of a specific joint. Facial motion is parameterized with facial animation parameters (FAPs), which either control movement of individual facial features or define the current facial expression and active visemes. Animations are applied to the character as streams of face and body animation parameters (FBAPs) values. When multiple animations affect the same character, they are blended together using linear interpolation.

RealActor provides several animation control features which are required for real-time interactive systems. *Behavior Tracker* (Fig. 4) is a component which contains data about the realized animations. During realization process, it uses timings of running animations and internal timing of the FAPlayer to create the realization feedback. The feedback is then passed to the Scheduler and used to remove animations that are finished with execution. The realization feedback is also provided through RealActor's API.

Furthermore, RealActor realizes multimodal utterances from BML scripts which are grouped into three different priorities: normal, merged and high priority. Normal behaviors have default priority, which is executed sequentially following the list of BML blocks. Merged behaviors are realized in addition to normal behaviors and can be used as responses to environment events, such as gazing at a person who entered the room during user-ECA interaction. High-priority behaviors interrupt running ones. When they are done executing, RealActor continues to run the interrupted behaviors. Examples of these behaviors are interruptions that do not necessitate behavior re-planning.

In RealActor we also implement functions for pausing, stopping and deleting behaviors which are running or queued. The functions are implemented using mechanisms from visage|SDK character animation framework.

5 Neural networks for behavior alignment

In this section we describe the approach we used to solve the issue of aligning animations with synthesized text, which is introduced in Section 3.2.

The idea of using machine learning techniques to extract prosodic features from speech is not new. Existing TTS engines often use Bayesian networks, Hidden Markov Models (HMM) and Classification and Regression Trees (CART) to estimate phoneme durations in real-time. However, the drawback of most of these approaches is that they are language-dependent due to their reliance on phonemes. Our approach instead relies on words as input and is applicable to any language.

In RealActor back-propagation neural networks (BNNs) [43] are used to estimate word duration and align them with animation in real-time. For that purpose, we had to design the layout of our neural network system and train the BNNs with a database containing word-duration pairs in order to achieve the network configuration which yields optimal synchronization results. We specified the following requirements for our system:

– When determining word duration, TTS does word segmentation on phoneme level. Estimated durations of these phonemes depend on their positions inside the word.

Segmentation and duration are language-dependent. Since we aimed to provide a solution applicable to any language, we decided to use simple segmentation of words into letters and train networks to use correlations between letters to estimate word durations.

- As there are many combinations of letters within words and words have different numbers of letters, multiple BNNs would be defined, specializing in words with a specific number of letters.
- BNNs produce output in 0–1 range. Therefore, normalization would be used to scale this output to word durations.
- Letters' ASCII code would be used as BNN input. Letter capitalization would be ignored.

To satisfy these requirements, we experimented with several system layouts. First, we designed a system of six BNNs, each specializing in words of a particular length. The networks were trained with a database containing 1.646 different words. Training was an iterative process, where letters of each word were coded and processed by an NN. The output duration was compared with actual word duration and NN weights subsequently adjusted depending on the estimation error. The process continued until the error dropped below the specified threshold. Once the training was complete, we evaluated the resulting BNN configurations using a set of sample sentences and words. The set contained 359 words, which differed from those used in the training process. While analyzing the results we observed that considerable estimation error occurred for words which were immediately followed by punctuation. This is due to the fact that these words take about 100 ms longer to pronounce than words without punctuation (which we henceforth refer to as plain words). To address this issue, we created another experimental layout, which contained 5 pairs of BNNs. Each BNN pair specialized in words of a specific length. However, the first BNN of a pair was trained to handle plain words, while the second one specialized in words followed by punctuation. Upon training and evaluating the BNNs, we found that they were indeed more effective in estimating word durations than the original setup. Therefore, we decided to use the latter approach in our final design.

For our final system we prepared 16 pairs of neural networks (NNs) and trained them to estimate word durations. Each of the 16 pairs was trained to handle words of a specific length (1–16), like in our experimental design. The remaining three pairs would handle long words of varying lengths (16–20 for the first pair, 20–25 for the second pair, 25–35 for the third pair). The words used for training were extracted from a variety of different contexts, including newspaper articles, film dialogue and literary fiction. The resulting database contained 9.541 words of varying lengths and meanings. The longest word encountered had 16 letters, so in the end we discarded the three pairs of NNs intended to handle long words, leaving a total of 32 BNNs in the system.

Evaluation Once the BNNs were prepared, we evaluated the effectiveness of our method by using the BNNs to estimate the time when an animation should start so that its synchronization point is aligned with a specific word in a sentence. Sample sentences and words used for evaluation differed from those used in the training process. The evaluation was done by summing up the predicted durations of words preceding the synchronization word to compute the predicted time interval before synchronization point and comparing it with the correct time interval to determine estimation error. Time intervals used in the evaluation were 500 ms and 1500 ms, corresponding to shortest and longest start-stroke durations for hand gestures, respectively. During evaluation we also took into account the fact that the human eye can tolerate deviations between speech and image of up to 80 ms [48], meaning that multimodal alignment error under 80 ms is unnoticeable to the viewer. The results of the evaluation are presented in Table 1.

As shown in the table, our system was able to align 92,26% of words for the short time interval and 73,26% of words for the long interval with alignment error not exceeding the 80 ms threshold. Furthermore, the system achieved 79,03% and 56,59% alignment rates with no measurable alignment error.

Support for other TTS engines Though RealActor uses a text-to-speech system based on Microsoft SAPI by default, it can be integrated with any TTS engine with minimal effort on the user's part. In order for multimodal synchronization to work correctly, the neural networks need to be retrained with the new TTS engine, which is a largely automatic process that can take from several hours up to a day, depending on the user's system.

6 Face animation model

In RealActor facial motion synthesis is realized using BML element "face" and attribute values which control movements of the eyes, eyebrows and mouth separately. Alternatively, face movements can be defined with Action Units (AUs) which control our face animation model.

6.1 Implementation approach

Ekman's Facial Action Coding System (FACS) [14] is designed for a trained human expert to detect the change of facial appearance. FACS defines in total 44 different Action Units (AUs), which describe the movements of facial muscles. Using AUs a trained human can decode any facial expression as a combination of different AUs with variable duration and intensity. Similarly, any facial expression can be synthesized as a combination of action units.

We model each AU as a procedural animation which controls the specific group of FAPs over a certain amount of time. Dynamics of each AU depend on its duration and intensity. During the behavior realization process, a central unit named AU Manager coordinates execution of several AUs using weighted blending and linear interpolation (Fig. 4).

Each *Action Unit* is based on the Attack-Decay-Sustain-Release model (ADSR) [11], which introduces animation of linear facial features points though following realization phases: *attack,*

Table 1 Neural network evaluation results

Time interval	500 ms	1500 ms
No. of words handled	310	258
No. of precisely aligned words	245	146
Alignment rate (error under 80 ms)	92,26%	73,26%
Alignment rate (no error)	79,03%	56,59%
Largest deviation	329 ms	463 ms
Smallest deviation	1 ms	2 ms
Average deviation	84.37 ms	126.00 ms

sustain, decay, release. While the BML defines behavior timings relatively using references to other elements, in our model a developer can alternatively specify duration of each phase absolutely. There are two reasons for this. First, the dynamics of face movement are highly important [55] and this approach provides greater control than BML itself. Second, by using explicit specification of phase durations, videos showing facial expressions can be easily re-created (an example is shown in the first case study of the evaluation). Accordingly, we propose an additional face attribute for the BML face element which controls timings of Action Units.

<face id ="f1" type="AU1" intensity="A" **au_duration**="400:500:800"/>

The above example defines a BML element with id "f1". F1 defines Action Unit 1 (which is *inner eyebrow raise*) of very low intensity. To control its dynamics we use *au_duration* attribute, which specifies that it performs very slowly, since AU1 amplitude is very low. The unit lasts 500 milliseconds and it takes 800 milliseconds until facial appearance goes back to neutral position, if there are no other facial actions. If not specified, default durations for attack, sustain, decay phases are 450, 800, 300 respectively for all Action Units.

Single Action Unit is modeled by analyzing results of face tracking. The tracker that we use has been developed at Linköpings University [24] and comes as part of visage|SDK framework. It applies manually adaptable textured Candide 3D model to a real human's face and then tracks a set of significant feature points and extracts global parameters—rotation and translation, as well as dynamic facial parameters (expressions) for each frame of the tracked video. For Action Unit modeling we adapt the face tracker to obtain the results of videos that accompany the book "Facial Action Coding System" [14]. The idea is to analyze how movements of specific feature points occur and apply the findings to MPEG-4 FAPS. For that purpose, we have modified the tracker's source code to save the global parameters of specific feature points in given files. We separately tracked each video showing a professional actor perform a single AU of certain intensity. First, texture of the Candide model with feature points (FPs) is applied to the actor's face (Fig. 5) and when the tracking is done, results are saved into a file. In that way we have saved results of videos of 34 different Action Units. Unfortunately, these videos display only the attack phase of the Action Unit.

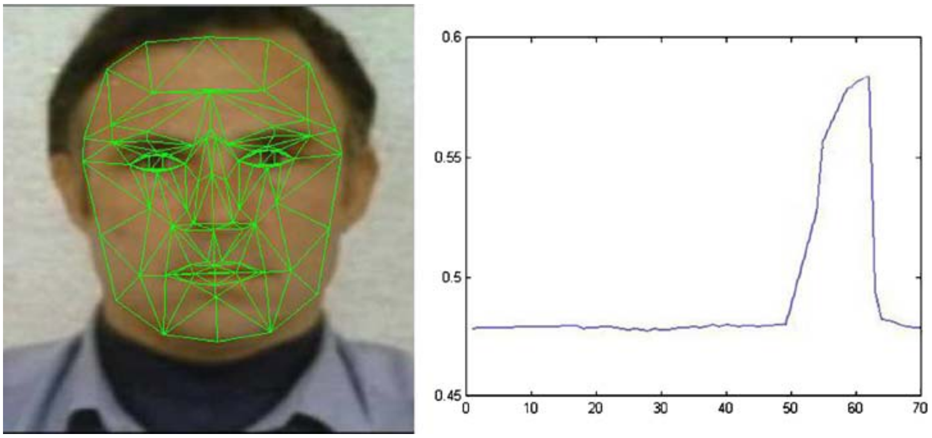

Fig. 5 (*left*) Face tracking system with adaptive mask. (*right*) Graph which shows movements of a left inner brow in 2D x-y plane

Values of global parameters of specific feature points over time are then displayed with MATLAB's simulation tool [33]. One graph displays the dynamics of movement of one feature point in 2D plane (usually X-Y space). For each Action Unit we create up to four graphs, depending on how complex the Action Unit is and what kind of movements it affects. For example, the lip movements move a group of several feature points (MPEG-4 FBA defines 12 FAPs that affect the lips) and to model the AU which affects lips we create and analyze four graphs.

With the help of Matlab's toolbox for the linear curve fitting graph, we then approximate the graphs with mathematical functions. Each approximation is then applied to a specific MPEG-4 FAP. The maximum amplitude of FAPs for each modeled AU is defined experimentally. While FACS defines five levels of intensity for each AU, in our model amplitude of FAPs is set linearly according to the maximum value and controlled intensity of a specific AU.

For example, we explain how we model AU1, described as "inner brow movement". Figure 5 depicts how the feature point of a left inner brow moves in a tracked video that shows an actor performing AU1. We approximate this function with the following equation (combination of linear and polynomial function):

$$\begin{array}{l} 0 < t < attack/2 f(t) = \frac{2}{3} * A_{max} * t \\ \frac{attack}{2} < t < attack f(t) = \frac{-2}{3 * attack^2} A_{max} * t^2 + \frac{4}{3 * attack} A_{max} * t \end{array}$$

The attack phase is divided into two subphases; linear and polynomial functions affect the final position of the feature point depending on the realization time. The function is then applied to FAPs *raise_r_i_eyebrow* and *raise_l_i_eyebrow* in a procedural action which implements AU1. Maximum value of amplitude *A* is defined after series of experimental movements of FAPs *raise_r_i_eyebrow* and *raise_l_i_eyebrow* and is set to 200. Sustain phase keeps feature point values the same, while decay phase uses linear fall to the neutral face position.

If a developer wants an ECA to realize AU1, they can control the dynamics of execution using *intensity* of AU1 and duration of *attack, sustain* and *decay* phases. Default intensity is the smallest one—"A"—and for AU1 its value is 40. Default duration for attack, sustain, decay phases is 450, 800, 300 respectively for all Action Units.

Using the modified face tracking system and results of analysis 34 Action Units were modeled in total. Most of the analyzed movements could be described with linear and polynomial functions. There were some graphs with peak undulations that probably come from bad video quality and these spikes were not taken into account in the approximation. In AUs which affect the mouth movement, only key feature points (lip corners and lip top and bottom) were analyzed and applied to MPEG-4 parameters. Since the tracker we used does not track eye openness, we modeled AUs which affect the eyes manually. Head movements were not analyzed since RealActor implements a head controller with head movements modeled with a similar approach [Brkic et al., 2008].

After animation implementation, modeled Action Units are compared to original video examples. Subjective evaluation shows good results, however limitations of our face model and MPEG-4 FBA standard for some AUs gives animations which cannot be the same as in the original video. First, our face model has no wrinkles. Second, some AUs describe face movements that cannot be animated using MPEG-4 FAPs. Examples are: lip points on Z plane, the movement significant for AU29; inner lip points on X plane, the movement significant for AU30, and nose on Y plane, the movement significant for AU9.

In real time facial motion synthesis, modeled AUs need to be coordinated if two same AUs happen simultaneously or two AUs affect the same FAPs. Therefore we design *AUManager,* which is a central face animation controller for blending Action Units. For each frame of animation the blender iterates through running Action Units to determine the final value of each FAP. The final value is chosen depending on how many (same or other type) AUs affect the same FAP using weighted blending. This makes an effect of natural transitioning between Unit phases and continuous repeats of AU. For example, when AU1 is in *decay* phase and new AU1 appears, the results of *decay* and *attack* phases will be blended, second AU1 will continue without moving the brows back to neutral position. Visually, an ECA raises his/her inner eyebrows, and just as they start falling down they raise back again smoothly.

6.2 Evaluating the face animation model

We conducted an empirical study to evaluate the effectiveness of our face animation model. The study consists of two cases:

1. Imitation of face movements.
2. Emotion synthesis and recognition.

For both studies we recruited 28 subjects, 13 males and 15 females, aged between 23 and 35, with no experience in character animation and emotion recognition. The study included subjective judgment of animation quality and recognition of synthesized face expressions.

Imitation of face movements In the first study we wanted to estimate the accuracy of our face model. In an ideal situation we would have recruited trained FACS coders to decode designed animations. Due to lack of sufficient subjects, we chose the following methodology:

- We manually annotated 6 videos of a human who is performing combinations of different AUs. Each video lasted 2 seconds.
- Using time annotation we designed 6 canned animations of an ECA imitating the human.
- Pairs of videos were shown to human subjects. They had to rate perceived quality of "imitation", i.e. how faithful movements of ECA's face to face movements of the human. The rating was done on a 5 points Likerts scale (0–4).

Evaluation results are shown in Table 2. Video 1 and video 6 were perceived as best quality. The second video which shows a subject wrinkling with the inner part of his eyebrows (AU4) and dilating his pupils (AU5) has a weak average grade. We suspect it's because participants were more affected by the wrinkles than by pupil dilation. Regarding the third video, combination of AU10 and AU12 in the original video makes a quadratic shape of the mouth. Using MPEG-4 FAPs we could not accomplish identical results. In summary, we are satisfied with the evaluation results.

Emotion synthesis and recognition For the second study we constructed animations of basic emotions with FACS according to theoretical specifications [14]. In total we made 8 canned animations (Fig. 6, Table 3). Each animation is scripted with BML and provided in RealActor's script library.

Table 2 Case study 1 evaluation results

Combination of AU	Average Quality (max 4)	Standard deviation
Video pair 1 (AU1 + AU2 + AU4)	3	0,902
Video pair 2 (AU4 + AU5)	2,07	1,184
Video pair 3 (AU10 + AU12 + AU6)	2,11	0,916
Video pair 4 (AU12 + AU27)	2,89	0,916
Video pair 5 (AU14 + AU23)	2,64	1,311
Video pair 6 (AU15 + AU17)	3,17	0,819

Animations were shown to evaluation subjects. After viewing each video, they had to report perceived emotions. They could choose one of the following replies: anger, disgust, joy, fear, surprise, sadness and uncertain. Evaluation results are shown in Table 4.

Joy, surprise, sadness were the most perceived emotions, which is similar to results of Gosselin and Kiroac [19]. Disgust and fear were perceived badly, as well as anger, which was also reported as sadness. We observed that fear was mistaken for surprise, which is a well-known phenomenon in psychology [12].

In summary, the animations designed for joy, surprise, and sadness were well recognized. The results have also shown that we need to improve animations of anger, fear and disgust.

7 Discussion and future work

Our research strives to create believable and expressive virtual characters in order to enhance the communication abilities of machines. In the paper we discuss multimodal

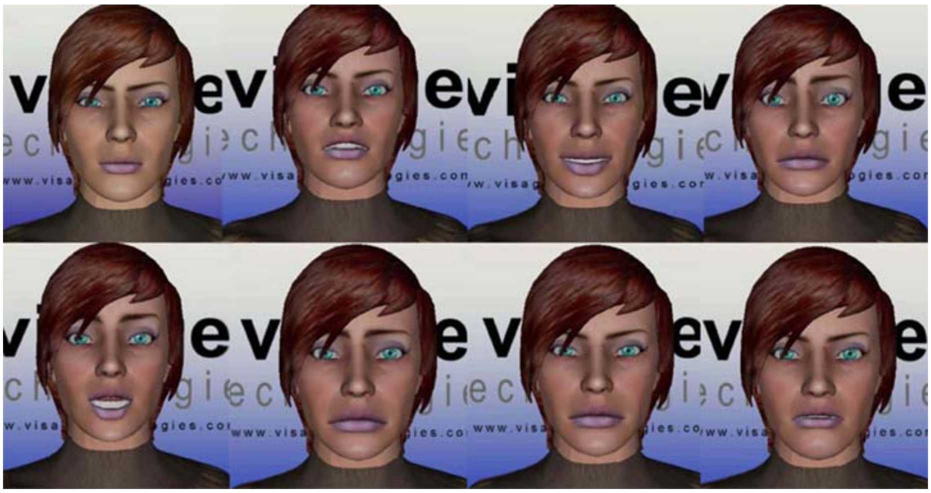

Fig. 6 Animations of facial expressions which were shown to evaluation subjects: anger, disgust1, joy, fear, surprise, sadness1, sadness2 and disgust2

Table 3 Designed animations and combination of Action Units

Animation	Emotion	Combination of AUs
1	Anger	4, 5, 7 and 27
2	Disgust	10 and 17
3	Joy	6 and 12
4	Fear	1, 2, 5, 4 and 15
5	Surprise	1, 2, 5 and 26
6	Sadness	1, 4, 11 and 15
7	Sadness	6 and 15
8	Disgust	11 and 17

synchronization issues for ECAs and present RealActor, an open-source character animation and behavior system with support for modeling multimodal communication. The system is based on Behavior Markup Language (BML) and freely available to the research community. To our knowledge, it is one of four BML-compliant animation systems in existence. We propose a universal, language- and engine-independent solution to the issue of synchronizing verbal and nonverbal expressions by using neural networks to estimate word durations. We develop and evaluate the face animation model based on Ekman's Action Units which can synthesize an infinite number of facial expressions. Our system is designed in such a manner that it can be integrated with existing engines and application frameworks with minimal effort. RealActor will serve as a basis for our multi-party tour guide system [7], which will be a true test of its capabilities.

While testing RealActor, we observed that manual authoring of BML behavior scripts can be a time-consuming and counter-intuitive process that requires a strong understanding of the BML specification and constant animation lexicon look-ups. One idea is to develop a graphical BML tool for more intuitive script authoring, while another is to provide a higher-level component for automatic behavior generation. Considerable research has been done on methods of automatically deriving multimodal behaviors from text and our approach would build upon work presented in [32]. The situation is similar with Action Units. Facial expression synthesis with our model requires a basic knowledge of Ekman's Facial Action Coding System (FACS).

Table 4 Case study 2 Evaluation results

Animations	Anger	Disgust	Joy	Fear	Surprise	Sadness	Not sure
Anger	**0,32**	0,07	0	0,07	0	0,20	0,32
Disgust1	0,29	**0,57**	0	0	0,04	0	0,10
Joy	0	0	**1**	0	0	0	0
Fear	0	0	0	**0,29**	0,46	0,21	0,04
Surprise	0	0	0	0,07	**0,89**	0	0,04
Sadness1	0	0,1	0	0	0	**0,79**	0,11
Sadness2	0	0,11	0	0,11	0	**0,57**	0,2
Disgust2	0,25	**0,46**	0,04	0,07	0	0,18	0

Although our system provides basic facial expressions scripted with BML code, we are contemplating development of a facial animation tool based on AU images and examples which would be an extension of the RealActor system. The tool would help researchers easily create facial expressions and script them with BML code realized by RealActor.

On low level, we plan to introduce a new animation system based on parametric motion graphs. The new system would support example-based parametric motion (achieved by blending multiple motion capture animations with the help of registration curves) [31] and employ parametric motion graphs for transitioning between motions [21]. In addition to this, we must improve our methods of gaze control, as constant gaze shifts play an important role in applications that involve multi-party interaction (such as the aforementioned tour guide system).

The current system provides full body animation with the exception of posture shifts. Since realization of body postures based on affective states is a frequently discussed topic in HCI literature [2, 10, 28], we find it essential to develop an affective posture controller for RealActor.

References

1. Albrecht I, Haber J, Peter Seidel H (2002) Automatic generation of nonverbal facial expressions from speech. In: In Proc. Computer Graphics International 2002, pp 283–293
2. Bianchi-Berthouze N, Kleinsmith A (2003) A categorical approach to affective gesture recognition. Connect Sci 15(4):259–269
3. BML Specification http://wiki.mindmakers.org/projects:BML:main
4. Brkic M, Smid K, Pejsa T, Pandzic IS (2008) Towards natural head movement of autonomous speaker agent. In: Proceedings of the 12th International Conference on Knowledge-Based Intelligent Information and Engineering Systems KES 2008. 5178:73–80
5. Cassell J (2000) Embodied conversational agents. The MIT (April 2000)
6. Cassell J, Vilhjalmsson HH, Bickmore T (2001) Beat: the behavior expression animation toolkit. In: SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques, ACM, pp 477–486
7. Cerekovic A, Huang HH, Furukawa T, Yamaoka Y, Pandzic, IS, Nishida T, Nakano Y (2009) Implementing a multiuser tour guide system with an embodied conversational agent, International Conference on Active Media Technology (AMT2009), Beijin, China, October 22–24, (2009)
8. Cerekovic A, Pejsa T, Pandzic IS (2010) A controller-based animation system for synchronizing and realizing human-like conversational behaviors, Proceedings of COST Action 2102 International School Dublin
9. Chovil N (1991) Discourse-oriented facial displays in conversation. Res Lang Soc Interact 25:163–194
10. Coulson M (2004) Attributing emotion to static body postures: recognition accuracy, confusions, and viewpoint dependence. J Nonverbal Behav 28(2):117–139
11. Dariouch B, Ech Chafai N, Mancini M, Pelachaud C (2004) Tools to Create Individual ECAs, Workshop Humaine, Santorini, September (2004)
12. Ekman P (1973) Cross-cultural studies of facial expression, pp 169–222 in P. Ekman (ed.) Darwin and Facial Expression
13. Ekman P (1979) In: about brows: emotional and conversational signals. Cambridge University Press, Cambridge, pp 169–202
14. Ekman P, Friesen W (1978) Facial action coding system: a technique for the measurement of facial movement. Consulting Psychologists, Palo Alto
15. Face Gen 3d Human Faces: http://www.facegen.com/
16. Foster ME (2007) Enhancing human-computer interaction with embodied conversational agents, Universal access in human-computer interaction. Ambient Interaction, ISSN 0302–9743, Springer Verlag
17. Fratarcangeli M, Adolfi M, Stankovic K, Pandzic IS (2009) Animatable face models from uncalibrated input features. In: Proceedings of the 10th International Conference on Telecommunications ConTEL

18. Gebhard P, Schröder M, Charfuelan M, Endres C, Kipp M, Pammi S, Rumpler M, Türk O (2008) IDEAS4Games: building expressive virtual characters for computer games. In Proceedings of the 8th international Conference on intelligent Virtual Agents (Tokyo, Japan, September 01–03, 2008). H. Prendinger, J. Lester, and M. Ishizuka, Eds. Lecture Notes In Artificial Intelligence, vol. 5208. Springer-Verlag, Berlin, Heidelberg, 426–440
19. Gosselin P (1995) Kirouac, Gilles, Le decodage de prototypes emotionnels faciaux, Canadian Journal of Experimental Psychology, pp 313–329
20. Hartmann B, Mancini M, Pelachaud C (2002) Formational parameters and adaptive prototype instantiation for MPEG-4 compliant gesture synthesis. In: Proc. Computer Animation. (19–21), pp 111–119
21. Heck R, Gleicher M (2007) Parametric motion graphs. In: I3D '07: Proceedings of the 2007 symposium on Interactive 3D graphics and games, New York, NY, USA, ACM (2007) 129–136
22. Heloir A, Kipp M (2009) EMBR—A realtime animation engine for interactive embodied agents. IVA 2009, 393–404
23. Horde3D - Next-Generation Graphics Engine, http://www.horde3d.org/
24. Ingemars N (2007) A feature based face tracker using extended Kalman filtering, 2007, University essay from Linköpings universitet
25. Irrlicht Engine, http://irrlicht.sourceforge.net/
26. Johnston M, Bangalore S (2000) Finite-state multimodal parsing and understanding. In Proceedings of the 18th Conference on Computational Linguistics - Volume 1 (Saarbrücken, Germany, July 31–August 04, 2000). International Conference On Computational Linguistics. Association for Computational Linguistics, Morristown, NJ, 369–375
27. Johnston M, Cohen PR, McGee D, Oviatt SL, Pittman JA, Smith I (1997) Unification-based multimodal integration. In Proceedings of the Eighth Conference on European Chapter of the Association For Computational Linguistics (Madrid, Spain, July 07–12, 1997). European Chapter Meeting of the ACL. Association for Computational Linguistics, Morristown, NJ, 281–288
28. Kleinsmith A, Bianchi-Berthouze N (2007) Recognizing affective dimensions from body posture. ACII (2007) 48–58
29. Kopp S, Wachsmuth I (2004) Synthesizing multimodal utterances for conversational agents. Comp Animat and Virtual Worlds 15:39–52
30. Kopp S, Krenn B, Marsella S, Marshall A, Pelachaud C, Pirker H, Thorisson K, Vilhjalmsson H (2006) Towards a common framework for multimodal generation: the behavior markup language. In: Intelligent Virtual Agents, pp 205–217
31. Kovar L (2004) Automated methods for data-driven synthesis of realistic and controllable human motion. PhD thesis, University of Wisconsin-Madison
32. Lee J, Marsella S (2006) Nonverbal behavior generator for embodied conversational agents. In: Intelligent Virtual Agents, pp 243–255
33. Matlab http://www.mathworks.com
34. McNeill D (1992) Hand and mind: what gestures reveal about thought. University of Chicago Press
35. Microsoft Speech API: http://www.microsoft.com/speech
36. Neff M, Kipp M, Albrecht I, Seidel HP (2008) Gesture modeling and animation based on a probabilistic re-creation of speaker style. ACM Trans Graph 27(1):1–24
37. OGRE - Open Source 3D Graphics Engine, http://www.ogre3d.org/
38. Oviatt SL, DeAngeli A, Kuhn K (1997) Integration and synchronization of input modes during multimodal human-computer interaction. In Proceedings of the Conference on Human Factors in Computing Systems: CHI '97, pages 415–422, Atlanta, Georgia. ACM Press, New York.
39. Pandzic IS, Forchheimer R (2002) MPEG-4 Facial Animation—The standard, implementations and applications", John Wiley & Sons (2002) ISBN 0-470-84465-5
40. Pandzic IS, Ahlberg J, Wzorek M, Rudol P, Mosmondor M (2003) Faces everywhere: towards ubiquitous production and delivery of face animation. In: Proceedings of the 2nd International Conference on Mobile and Ubiquitous Multimedia MUM 2003, pp 49–55
41. Pejsa T, Pandzic IS (2009) Architecture of an animation system for human characters. In: Proceedings of the 10th International Conference on Telecommunications ConTEL 2009
42. Pelachaud C (2009) Studies on gesture expressivity for a virtual agent. Speech Communication, special issue in honor of Bjorn Granstrom and Rolf Carlson, to appear
43. Rojas R (1996) Neural networks—a systematic introduction. Springer-Verlag
44. Schroeder M, Hunecke A (2007) Mary tts participation in the blizzard challenge 2007. In: Proceedings of the Blizzard Challenge 2007

45. Smid K, Zoric G, Pandzic IS (2006) [huge]: Universal architecture for statistically based human gesturing. In: Proceedings of the 6th International Conference on Intelligent Virtual Agents IVA 2006, pp 256–269
46. Spierling U (2005) Interactive digital storytelling: towards a hybrid conceptual approach. Paper presented at DIGRA 2005, Simon Fraser University, Burnaby, BC, Canada
47. Spierling U (2005) Beyond virtual tutors: semi-autonomous characters as learning companions. In ACM SIGGRAPH 2005 Educators Program (Los Angeles, California, July 31–August 04, 2005). P. Beckmann-Wells, Ed. SIGGRAPH '05. ACM, New York, NY, 5
48. Steinmetz R (1996) Human perception of jitter and media synchronization. IEEE J Sel Areas Commun 14(1)
49. Stone M, DeCarlo D, Oh I, Rodriguez C, Stere A, Lees A, Bregler C (2004) Speaking with hands: Creating animated conversational characters from recordings of human performance. In: Proceedings of ACM SIGGRAPH 2004 23:506–513
50. Taylor PA, Black A, Caley R (1998) The architecture of the festival speech synthesis system. In: The Third ESCA Workshop in Speech Synthesis, pp 147–151
51. Thiebaux M, Marshall A, Marsella S, Kallmann M (2008) Smartbody: behavior realization for embodied conversational agents. In: Proceedings of Autonomous Agents and Multi-Agent Systems AAMAS
52. Van Deemter K, Krenn B, Piwek P, Klesen M, Schroeder M, Baumann S (2008) Fully generated scripted dialogue for embodied agents. Articial Intelligence, pp 1219–1244
53. Vilhjalmsson H, Cantelmo N, Cassell J, Chafai NE, Kipp M, Kopp S, Mancini M, Marsella S, Marshall AN, Pelachaud C, Ruttkay Z, Thorisson KR, Welbergen H, Werf RJ (2007) The behavior markup language: recent developments and challenges. In: IVA '07: Proceedings of the 7th international conference on Intelligent Virtual Agents, Springer-Verlag, pp 99–11
54. Vinayagamoorthy V, Gillies M, Steed A, Tanguy E, Pan X, Loscos C, Slater M (2006) Building expression into virtual characters. In Eurographics Conference State of the Art Reports
55. Wehrle T, Kaiser S, Schmidt S, Scherer KR (2000) Studying the dynamics of emotional expression using synthesized facial muscle movements. J Pers Soc Psychol 78(1):105–119
56. Zorić G, Pandžić IS (2005) A real-time lip sync system using a genetic algorithm for automatic neural network configuration, in Proceedings of the International Conference on Multimedia & Expo, ICME 2005, Amsterdam, Netherlands
57. Zoric G, Smid K, Pandzic IS (2009) Towards facial gestures generation by speech signal analysis using huge architecture. In: Multimodal signals: cognitive and algorithmic issues: COST Action 2102 and euCognition International School Vietri sul Mare, Italy, April 21–26, 2008 Revised Selected and Invited Papers, Berlin, Heidelberg, Springer-Verlag, pp 112–120

Aleksandra Čereković is a Ph.D. student and research and teaching assistant at the Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia. As undergraduate student, she has been working on the topic of lip synchronization for real-time character animation. In year 2006 she had participated in eNTERFACE '06 workshop on Multimodal Interfaces on the project "An Agent Based Multicultural Customer Service Application". Two years later she was a leader of the project "Multimodal communication with a tour guide ECA" on the eNTERFACE workshop in Paris. Besides, she had participated in Peach Summer School in year 2007, and two spring schools of the COST 2102 action in Vietri and Dublin (year 2007 and 2009). Her research interests are multimodal communication, virtual character animation and embodied conversational agents.

Igor S. Pandžić is an Associate Professor at the Department of Telecommunications, Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia and director of Human-Oriented Technologies Laboratory (HOTLab). He teaches undergraduate and postgraduate courses in the fields of virtual environments and communications. His main research interests are in the field of computer graphics and virtual environments, with particular focus on character animation, embodied conversational agents, and their applications in networked and mobile environments. Igor also worked on networked collaborative virtual environments, computer generated film production and parallel computing. He published four books and around 90 papers on these topics. Formerly he worked as a Senior Assistant at MIRALab, University of Geneva, Switzerland, where he obtained his PhD in 1998. The same year he worked as visiting scientist at AT&T Labs, USA. In 2001–2002 he was a visiting scientist in the Image Coding Group at the University of Linköping, Sweden, and in 2005 at the Department of Intelligence Science and Technology, Kyoto University, on a Fellowship awarded by Japan Society for Promotion of Science. He received his BSc degree in Electrical Engineering from the University of Zagreb in 1993, and MSc degrees from the Swiss Federal Institute of Technology (EPFL) and the University of Geneva in 1994 and 1995, respectively. Igor was one of the key contributors to the Facial Animation specification in the MPEG-4 International Standard for which he received an ISO Certificate of Appreciation in 2000. He held various functions in organizing numerous international conferences and served as guest editor for a special topic in IEEE Communications Magazine. He has been active in international research collaborations within the EU research programmes since the 3rd Framework Programme, as well as in national research projects and collaboration with industry.