Speaking with Hands: Creating Animated Conversational Characters from Recordings of Human Performance

Matthew Stone, Rutgers
Doug DeCarlo, Rutgers
Insuk Oh, Rutgers
Christian Rodriguez, Rutgers
Adrian Stere, Rutgers
Alyssa Lees, NYU
Chris Bregler, NYU

Figure 1: "Ow, dude! You missed your cue to take hold of the board!" Clips of sound and clips of motion are selected from separate performances, spliced together and resynchronized to create the animated delivery of a meaningful new utterance.

Abstract

We describe a method for using a database of recorded speech and captured motion to create an animated conversational character. People's utterances are composed of short, clearly-delimited phrases; in each phrase, gesture and speech go together meaningfully and synchronize at a common point of maximum emphasis. We develop tools for collecting and managing performance data that exploit this structure. The tools help create scripts for performers, help annotate and segment performance data, and structure specific messages for characters to use within application contexts. Our animations then reproduce this structure. They recombine motion samples with new speech samples to recreate coherent phrases, and blend segments of speech and motion together phrase-by-phrase into extended utterances. By framing problems for utterance generation and synthesis so that they can draw closely on a talented performance, our techniques support the rapid construction of animated characters with rich and appropriate expression.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation; I.2.7 [Artificial Intelligence]: Natural Language Processing—Language Generation; Speech synthesis; J.5 [Arts and Humanities]: Performing Arts

Keywords: animation, motion capture, motion synthesis, language generation, conversational agents, speech synthesis

1 Introduction

Engaging dramatic performances convincingly depict people with genuine emotions and personality. This illusion springs not so much from an author's script as from the nuances of the performer's body and voice. By evoking nuances of spontaneous action, performers can animate characters' utterances, embody characters' identities, and portray the detail of characters' feelings.

It takes a talented artist to bring a script to life. Research on interactive characters increasingly focuses on example-based synthesis techniques that can directly exploit a performer's talent. For example, many methods for computer character animation now start from collections of motion data captured from human performances, as in [Arikan and Forsyth 2002; Kovar et al. 2002a; Lee et al. 2002a; Li et al. 2002; Pullen and Bregler 2002]. In fact, even human animators sometimes use an actor's rendition to help guide their work. Likewise, in automatic speech synthesis, some of the most effective current results are obtained using application-specific databases of recorded voice talent [Black and Lenzo 2000]. Despite the common perspective across these fields, researchers have not yet shown how to reconcile their different example-based synthesis techniques.

We present an integrated framework for creating interactive, embodied talking characters from human performances. We show how to collect, organize and manipulate sound and motion data to build a character engine that synthesizes expressive animated utterances directly from the state of an underlying application. Our integration depends on using constraints from human communication to organize the design, as advocated by [Cassell 2000]. In our results, such as that in Figure 1, and in the accompanying video, these constraints allow the meaning and personality in the actor's original recordings to resurface in our character's new utterances.

Our approach brings two key contributions. First, we show how to use a performer's intuitions about a character's behavior as an alternative to more comprehensive reasoning about communication. We use natural language generation techniques to link the performer's script to aspects of the application state. These links establish an application-specific semantics for communicative behaviors. Once we capture performance and link it to the application state, we no longer have to plan, annotate or infer the specific form and meaning that appears in a gesture. Annotators simply judge whether motions in the actor's performance are descriptive (iconic presentations of specific content) or expressive (metaphorical presentations of function, which are more common and can often be quite broadly reused). Thus our approach is distinguished from that of BEAT [Cassell et al. 2001], which relies on natural language un-

derstanding techniques to predict the underlying linguistic and discourse structure of an input text. Our approach handles expressive gestures with new generality.

Second, where previous approaches synthesize speech and motion separately, our approach selects and concatenates units of speech and units of motion as part of a single optimization process. Our units of speech are short phrases of a few words from the original performance. These intervals correspond to *intonation phrases*, which are the high-level units of natural face-to-face communication. Our units of motion are extracted from the corresponding temporal intervals, and can be warped to accompany different speech in a new utterance. In natural utterances, the peaks of emphasis in gesture occur simultaneously with peaks of emphasis in speech. Gesture transitions lie near the boundaries of phrases, away from the emphasis in speech and motion. We define an objective function over possible utterances that balances seamless speech, continuity of gesture, and consistency of speech and gesture. Optimizing this objective lets us trade off the units of speech and the units of motion that we include in an utterance, and search more flexibly for realizations that require minimal processing. This gives a more robust design that can use more of the available data. For example, it is no longer the case that the timing of gesture during speech must be driven entirely by a standalone speech synthesizer, as in BEAT [Cassell et al. 2001] and other previous work.

Our framework supports creative effort across the many steps required to design, capture and realize a new character. In particular, a single compositional, context-dependent description of system utterances guides content creation, data collection and utterance generation. In this framework, teams can develop interactive characters with just a modest increase over the effort of scriptwriting, data acquisition and motion computation already required to use performance data for canned character animation. The results have immediate application for entertainment, training, and conversational human-computer interaction generally.

2 Background and Related Work

2.1 Communication

In face-to-face dialogue, utterances consist of coordinated ensembles of coherent verbal and non-verbal actions [McNeill 1992; Bavelas and Chovil 2000; Engle 2000]. No modality is privileged or primary. The speech consists of a sequence of *intonation phrases*. Each intonation phrase is realized with fluid, continuous articulation and a single point of maximum emphasis. Boundaries between successive phrases are associated with perceived disjuncture and are marked in English with cues such as pitch movement, vowel lengthening and sometimes, but not always, pausing.[1] See [Pierrehumbert and Hirschberg 1990; Silverman et al. 1992]. Gestures are performed in units that coincide with these intonation phrases, and points of prominence in gestures also coincide with the emphasis in the concurrent speech [Ekman 1979; McNeill 1992].

People use these ensembles of gesture and speech to offer a consistent description of objects and events, but these ensembles need not respect the phrase boundaries found in traditional syntactic structure. Rather, they organize an utterance into units that describe a common topic and achieve a consistent communicative intention [McNeill 1992; Steedman 2000].

This model is increasingly corroborated by fine-grained experimental analysis of the delivery and interpretation of human utterances [McNeill et al. 2001; Krahmer et al. 2002; Cerrato and Skhiri 2003]. Correspondingly, systems for conversational animation since [Cassell et al. 1994] have aimed to coordinate the meaning and the realization of animated actions with the delivery of simultaneous speech. Our work continues this tradition.

[1] For readability, here, we use intonation phrase as a cover for two different kinds of high-level prosodic units, the intonation phrase and the intermediate phrase, distinguished in English phonology.

2.2 Speech Synthesis and Language Generation

The ability to access extensive databases of recorded speech has enabled speech synthesis systems to concatenate minimally-processed segments of recorded speech [Hunt and Black 1996]. The technology underlies high-quality wide coverage synthesis systems, such as AT&T's Next-Gen synthesizer [Beutnagel et al. 1999]. In fact, domain-specific synthesizers can use specific recordings of application utterances to recreate the rendition and voice quality of a human performer even more closely [Black and Lenzo 2000]. We build directly on these techniques here.

Building spoken utterances that express application data involves natural language generation (NLG) as well as speech synthesis (see [Reiter and Dale 2000]). Our work is also informed by prior work in NLG. In particular, we adopt a commonly-used template-based technique for NLG, which involves the recursive, context-dependent expansion of utterance structure as guided by application data, as in [Seneff 2002]. A particular advantage of this approach is the ability to generate a network of possible utterances for further processing as in [Langkilde 2000; Bangalore and Rambow 2000; Bulyko and Ostendorf 2002]. Template generation systems can also help to select and organize utterances for subsequent recording to streamline domain-specific speech synthesis [Theune and Klabbers 2001; Pan and Wang 2002].

2.3 Motion Synthesis

Performance-driven animation rests on technology for recording human motion and rendering it back just as performed; such technology has long embraced the expressive movements of human communication [Williams 1990]. Performance data can also serve as the basis for synthesizing new motion; we can warp captured motion in time and space [Witkin and Popović 1995], interpolate captured motion to vary its emotional character [Rose et al. 1998], retarget it to new characters [Gleicher 1998], and preserve its dynamics in the process [Popović and Witkin 1999]. Such manipulation remains limited in its ability to adapt the gross structure of captured motion. Using a more extensive database of captured motion can achieve more flexible synthesis while reducing the need for radical editing of performance data. See [Arikan and Forsyth 2002; Kovar et al. 2002a; Lee et al. 2002a; Li et al. 2002; Park et al. 2002; Pullen and Bregler 2002]. These approaches exploit large databases of motion segmented into short units that can be blended together. The approaches construct new motions by selecting sequences of units to splice together so as to optimize the transitions between them and to satisfy global constraints on motion. Since characters' gestures must match both the content and timing of simultaneous speech, two further refinements to these techniques prove vital for conversational animation. First, following [Arikan et al. 2003], we can classify motion data into qualitative categories and factor this information into the selection of units in motion synthesis. This lets us restrict our attention to gestures with particular patterns of form or content. Second, following [Kim et al. 2003], we can readily warp the timing of performance data to synchronize with external events. In our case, this lets us synchronize temporal emphasis in speech with the peak of effort in the accompanying movement.

2.4 Conversational Animation

Our work falls in a broad category of research which aims to develop animated characters known as *embodied conversational*

agents (ECAs). ECAs combine synthesized speech with hand gestures, head movements and facial displays to recreate the communicative functions and behaviors of human dialogue. Cassell et al. [2000] introduce and survey this work. ECAs need a wide range of expressive movements beyond the articulatory movements (e.g., of the lips) studied in research on *visual speech* [Schroeter et al. 2000; Ezzat et al. 2002; Beskow 2003]. ECAs also need to maintain fine-grained semantic and temporal connections between these movements and ongoing speech, which requires going beyond *behavioral* techniques that drive animation from statistical models [Perlin and Goldberg 1996; Brand 1999; Lee et al. 2002b]. Previous methods for using performance data have been sufficient for visual speech and behavioral animation, but not for ECAs.

ECAs typically involve a component that generates coordinated units of speech and gesture, and another that realizes the animation. The generation component sometimes adopts rule-based planning methods, as in [Cassell et al. 1994; Pelachaud et al. 1996], and sometimes reconstructs conversational actions from text input, as in [Cassell et al. 2001]. Our template-based generator differs from these techniques because it does not require a general approach to plan or fill in communicative behavior. Moreover, we use the templates to acquire data from a performer and to constrain data-driven synthesis, rather than to formulate specific input for domain-general synthesis as in [Stone and DeCarlo 2003; Bickmore 2003].

Procedural components for realizing conversational animation, including [Beskow et al. 2002; Kopp and Wachsmuth 2004; DeCarlo et al. 2002], share our emphasis on coordination of speech and gesture. They accept generic multimodal input specifications indicating the units of each utterance, the categories of motion to accompany each unit, and the points of synchrony between them. We explore motion capture as an alternative in order to enable extremely rapid development of high-quality application-specific ECAs. Of course, with performance-driven conversational animation, we cannot yet achieve the generality of procedural animation.

3 Our Approach

We approach character creation through a single end-to-end framework. As outlined in Figure 2, our framework ties together activities of content authoring and data preparation during the development of a conversational character, and run-time processes that use this content and data for generation and animation. The unified approach greatly streamlines character development. It makes it possible to capture the data needed for a character with a limited number of performances, to catalogue performance data with limited human effort, and to synthesize new utterances that best exploit the available data.

- *Content authoring*. As described in Section 3.1, a scriptwriter designs what the character will say. Automatic tools then compute the utterance units implicit in the specification, formulate a concise script for a performer, compute a database specification that organizes the anticipated sound and motion recordings, and compile an application-specific generator that will index into the resulting database.

- *Data preparation*. As we describe in Section 3.2, analysts can rely on automatic analysis to describe the possible form, content and function of motion. But they must still annotate the perceptually prominent moments of emphasis in speech and gesture and classify each gesture to characterize the kind of meaning it carries. This analysis is closely supported by automatic tools for speech processing and data visualization.

- *Generation and animation*. Finally, as we describe in Section 3.3, at run-time, our synthesis engine runs the generator with application input to formulate each utterance specification. The synthesizer recombines performance data to fit this specification. In particular, our engine solves a unified dynamic programming problem which simultaneously optimizes the selection of speech and motion units in pursuit of seamless speech, smooth motion, and consistent timing and content across speech and motion. We realize this sequence by adjusting pitch and volume to splice successive speech segments together, warping successive motions to blend them together, and retiming each motion to synchronize with the marked points of emphasis in the corresponding speech.

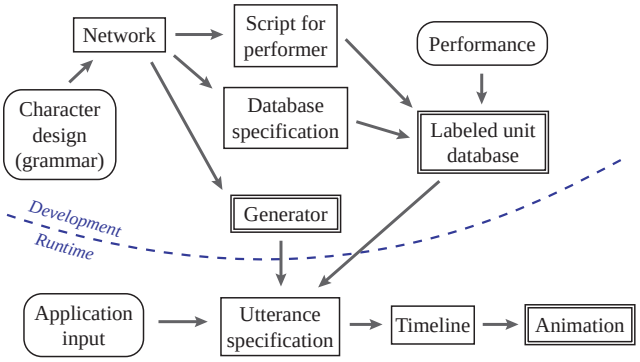

Figure 2: Designing an interactive conversational character from performance data. A single specification, the character design, is automatically transformed into a network representation that organizes the entire development process. We compute the performer's script from the network, use the network to characterize performance elements and set them up into a database, and compile the network into a generation engine. The database also includes descriptions for the timing and content of gesture which is annotated with machine assistance. At run-time, the application input, generation system and database together determine a space of possible utterance realizations. We optimize in this space to derive the timeline for an utterance, which is then animated by stitching together segments of motion and speech performance.

3.1 Authoring characters

In our framework, the generation of conversational action is an extension of the generation of spoken utterances. We have to characterize the organization of characters' utterances, describe the meaning and use of these utterances in the application and reason systematically about what the character can say. Our approach to character authoring therefore extends techniques for NLG.

The structure of natural language calls for authors to organize characters' communicative behaviors hierarchically into categories. Different expressions of the same category may be substituted for one another, which will change the meaning of the utterance but not its grammar. We use traditional phrase structure grammars to encode this information compactly and systematically. (See e.g. [Jurafsky and Martin 2000].) Figure 3 shows a small example specifying sixteen utterances for a character. The italicized symbols are nonterminals. The notation $A \rightarrow R$ indicates a production that replaces A with the string R during a derivation. No recursion is allowed. This rules out families of rules such as $A \rightarrow BC$ and $C \rightarrow AD$ that can expand any nonterminal (here A) into a string in which the nonterminal appears again. This restriction keeps the set of utterances finite and simplifies the processes of designing, recording, implementing, and validating a character.

The symbol ✠ is an instruction to include a prosodic boundary at a specified point within the utterance. Designers can use ✠ like

Start → *Frame Sentence*

Frame → on that run ✠
  IF: *contrastive*(*cxt*)
  AS: *contrast*

Frame → ε

Sentence → you *VerbPhr*

VerbPhr → *Modifier Property*

Modifier → still
  IF: *info*(*cxt*) = *prev-info*(*cxt*)
  AS: *reminder*

Modifier → ε

Property → *Predicate Action*
  IF: *type*(*cxt*) = *action-error*

Predicate → waited too long ✠
  IF: *error-type*(*cxt*) = *late*
  AS: *correct-error*

Predicate → were too quick ✠
  IF: *error-type*(*cxt*) = *early*
  AS: *correct-error*

Action → to grab the board ✠
  IF: *error-action*(*cxt*) = *grab*
  AS: *correct-action*

Action → to jump off ✠
  IF: *error-action*(*cxt*) = *jump*
  AS: *correct-action*

Figure 3: A small template grammar describing possible utterances such as, *on that run, you still were too quick to grab the board* and *you waited too long to jump off*. IF associates productions with a condition that tests the application context (*cxt*). AS associates productions with an abstract communicative function. ✠ marks a prosodic break.

a comma, to segment utterances into intonation phrases independently from their canonical syntactic structure. This represents a grammar that specifies prosodic markup for synthesized utterances, as in [Seneff 2002]. However, our phrasing does not provide input to an existing speech synthesizer. Instead, we will suggest this phrasing in the performer's script and use this phrasing to index the resulting recordings.

NLG systems can use a grammar in various ways, including translating an input semantic representation or planning mental effects on an audience [Reiter and Dale 2000; Cassell et al. 2000]. *Template NLG* is another alternative [Theune and Klabbers 2001; Bulyko and Ostendorf 2002; Seneff 2002]. Template NLG uses the grammar directly to construct utterances without formulating an intermediate communicative plan or semantic representation. In template NLG, nonterminals in a derivation are holes that need to be filled as a function of the current state of an interactive application. Productions in the grammar are rules for filling in the holes. Each production includes a *condition* that tests the context and says when it should be used.

Figure 3 is a template grammar. Its conditions test features of the variable *cxt*, the current application state. For example, one production specifies using the phrase "waited too long ✠" to characterize an error the user has made. The production is restricted by the condition *error-type*(*cxt*) = *late* that checks that the user has made this error. The grammar of Figure 3 also declares the abstract *communicative function* of phrases. This declaration summarizes why a phrase is included in the character's repertoire. For example, productions "waited too long ✠" and "were too quick ✠" share the function *correct-error*, because they spell out the kind of error the user made. By default, the alternative productions that rewrite each symbol are assumed to share a common function.

The specified grammar is automatically compiled into a transition network that can plan possible utterances by sequencing units of speech and gesture. The compilation implements the standard correspondence between finite grammars and state machines, and goes on to eliminate non-null transitions that do not end in ✠ by concatenating them with their successors [Hopcroft et al. 2000]. Each edge in the compiled network now groups together related words that should be uttered as a single phrase, regardless of syntactic constituency. Moreover, each edge associates a phrase with an application-specific condition for its use (using IF) and the application-specific communicative function it achieves (using AS). Figure 4 shows such a phrasal network for the grammar of Figure 3.

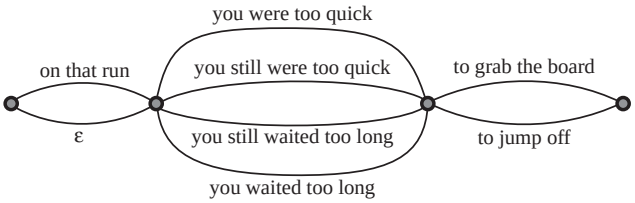

Figure 4: A network representation of the grammar of Figure 3, in which each edge encodes an instruction to use a complete intonation phrase; in our approach, such edges can abstract a single choice of coordinated speech and gesture.

- On that run ✠ you were too quick ✠ to grab the board ✠
- On that run ✠ you still were too quick ✠ to jump off ✠
- On that run ✠ you still waited too long ✠ to jump off ✠
- On that run ✠ you waited too long ✠ to grab the board ✠

Figure 5: A concise script for the network of Figure 4.

(The figure suppresses the attributes of condition and function associated with each edge.) Note how *parts* of productions in the grammar can combine together in edges such as "you still were too quick" to form intonation phrases that do not match the hierarchical grammatical structure. After compilation, we no longer need to consider the grammatical structure that makes up individual units. This edge achieves a combination of functions: *reminder* and *correct-error*. In other words, it is used to tell the user a type of error that they have just repeated. This edge is associated with the conjunction of the conditions *info*(*cxt*) = *prev-info*(*cxt*) and *type*(*cxt*) = *action-error* and *error-type*(*cxt*) = *early*. So the edge will be selected only when the current error was to act too early and the user made the same error in the previous interaction.

Network transformations have been used in NLG to allow dynamic programming tools to explore spaces of syntactic derivations and other realizations [Langkilde 2000; Bangalore and Rambow 2000; Bulyko and Ostendorf 2002]. Such methods typically apply during synthesis, to break down edges from high-level units like words and phrases into increasingly fine-grained units. Instead, we group units during development in order to conduct the synthesis more effectively with larger units.

In particular, we exploit the network representation to generate the performer's script automatically. Each line in the script is a path through the network—a possible complete utterance. As long as the script contains one performance for each edge, the character will be able to synthesize any path through the network. (Obtaining several renditions leads to a richer database with useful variability.) For the network of Figure 4, only four renditions are required to create sixteen possible utterances. Figure 5 offers a representative set. Note that performance sessions are most effective if variability is equally distributed within the utterance.

We can calculate the size of the script from the hierarchical structure of our network. When a subnetwork offers multiple paths from start to end, we need all the lines from each path. Along successive stages, we need only provide as many lines as the most variable stage. Thus Figure 4 needs four lines because of the four edges for the second phrase. We use precomputed counts to select and concatenate edges corresponding to numerically-indexed script lines. Alternative scripts can be generated by randomization.

3.2 Preparing performance data

The next step is to capture the performances from which to synthesize the utterances we have specified. We start from the automatically-compiled script. The performer interprets it to offer

her own embodied delivery with added expressive detail. These performances can flesh out the application content, for example using distinctive gestures. They can also portray the character's personality and potential relationship with the interlocutor, and so highlight phrases' communicative function. These added elements of performance are what subsequent processing must preserve.

To do so, we need to know what the performer did and when. Existing automatic tools for speech and motion data provide a good start. They can provide a temporal alignment of the performance with its transcript, extract acoustic and visual features, and prepare resources for visualizing and understanding the data.

However, human analysis remains necessary for accurately timing and categorizing the performer's behavior. Points of perceived prominence in speech and gesture correlate with automatically-extracted features but do not match them exactly. We annotate when they occur. We also annotate the perceived relationship between gesture and speech. In particular, we need to know whether the gesture is *descriptive* or *expressive*. Descriptive gestures elaborate the referential content of the utterance. This is typical of iconic gestures that represent objects or events in space. For example, a gesture that pantomimes the action of jumping while ongoing speech talked about a jump would count as a descriptive gesture. Descriptive gestures have to be indexed under the content associated with the performance. A convincing character cannot pantomime a jump while describing another action.

Expressive gestures highlight the attitude of the speaker towards what she is saying and comment on the relationship of speaker and addressee, which is typical of metaphorical and beat gestures. For example, a gesture in which the performer brings her hands toward her head and shakes them to pantomime a frustrated reaction should count as expressive. So should a gesture in which the performer points at the addressee and shakes her hand side-to-side in reprimand. Expressive gestures should be indexed under the function associated with the performance. Pantomimed frustration will fit any context that reports a frustrating outcome. An embodied reprimand fits any context where the character describes something the user has done wrong. For more on the meanings of different kinds of gesture, see [McNeill 1992; Bavelas and Chovil 2000].

Note that we do not describe the function and content of gesture independently as part of annotation. This principled decision reflects the complexity of gesture semantics. It is especially hard to label metaphorical gestures with a specific communicative function and then reason correctly about it. We assume that the content and function that the template generator associates with each performance unit already characterizes the gesture precisely enough for the application. We leave the rest to our skilled performer.

We found that it takes about two hours per minute of speech (about ten utterances in our examples) to check automatic alignment and prosodic features, and to extract points of emphasis and pitch. A further two hours per minute of video is required to check the coding and timing of gesture. It seems possible to improve this substantially using a more focused codebook, a more ergonomic coding environment, and of course better automatic tools.

Validation can help to minimize any remaining errors after human annotation. We can do this by checking that phrase data is appropriately resynthesized from annotations. We can use the synthesis techniques explored in Section 3.3, but procedural animation can be used as well.

3.3 Synthesis techniques

3.3.1 Unit selection

To plan a new utterance, the generator traverses the network from context information supplied by the external application. This determines the content and communicative function of each of the phrases the character needs to realize. To animate these phrases,

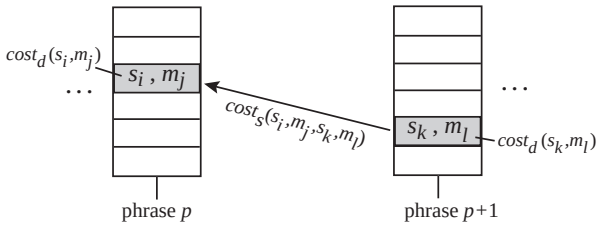

Figure 6: Dynamic programming solves simultaneously for combinations of speech and gesture.

we must pick suitable sound recordings and gesture performances. This is the *unit selection problem*.

Each edge traversed in the generator corresponds to a choice of sound and a choice of motion for realization. The options for each stage can be looked up in the database, giving a set $sound(p)$ of sound units and a set $motion(p)$ of motion units for each phrase p from 1 to the length n of the utterance. We write S_p for the sound to be presented at time p and M_p for the motion to be presented at time p, and use s_i and m_j to represent entries in the database. Our options for realization thus consist of a set U given by

$$\{\langle S_1 \ldots S_n, M_1 \ldots M_n \rangle | \forall p \in 1..n : S_p \in sound(p) \wedge M_p \in motion(p)\}$$

For realization, we want to pick the best element of U that we can.

We measure a sequence based on its *cost*, which serves as an approximate, heuristic measure of the perceptible degree to which a unit of performance will have be modified in a final realization. The overall form of the cost function is guided by previous work in example-based synthesis [Hunt and Black 1996; Arikan and Forsyth 2002; Kovar et al. 2002a], and by the computational advantage of obtaining a factored decomposition of cost. We define the cost functions as the sum of a number of quadratic penalty terms. (Differences are penalized directly to prefer values near 0; for a ratio r, we penalize $\log(r)$ to prefer values near 1.) The particular terms reflect the kinds of processing that we actually perform in the course of realizing utterances, as described in Section 3.3.2.

We assign $cost_d(s_i,m_j)$ to *delivering* each phrase, based on the match between its sound s_i and motion m_j. We chose $cost_d(s_i,m_j)$ to have one term which penalizes the amount of time-warping (a ratio) required to align the motion to the sound. We also assign $cost_s(s_i,m_j,s_k,m_l)$ to *splicing* the recordings between successive sounds s_i and s_k and motions m_j and m_l. Three terms are used in defining $cost_s(s_i,m_j,s_k,m_l)$. The first measures the difference in joint positions between m_j and m_l, temporally averaged across the short overlay window where corresponding samples will be interpolated. The second term penalizes differences in pitch (a ratio) between the peak of s_k and the peak that originally followed s_i. The third term penalizes blends that may require motion data from outside m_j and m_l taken from adjacent phrases in the original recording (which is safe because blending occurs only in the middle of utterances). Thus, this objective takes the form

$$\sum_{p=1}^{n} cost_d(S_p,M_p) + \sum_{p=1}^{n-1} cost_s(S_p,M_p,S_{p+1},M_{p+1})$$

The terms of this objective only involve adjacent segments. That means we can compute its minimum by dynamic programming—the best sequence S_p and M_p up to phrase p must include the best sequence up to time $p-1$ that ends in S_{p-1} and M_{p-1}.

A danger of this procedure is the complexity of computing the best options at time $p+1$ from the best options at time p. This grows *quadratically* in the number of states, and the number of states grows in turn as the *product* of the number of units available in each modality. There are two reasons why we can expect

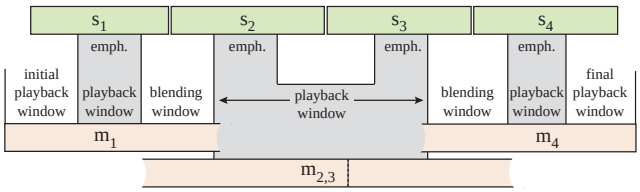

Figure 7: Points of emphasis in speech and points of continuity and discontinuity of motion determine the windows over which motion is blended together and warped across time.

such a computation to remain scalable. First, we are selecting *phrases* and *gestures* as whole units, so n is short and there are relatively few options for realizing each phrase. In effect, as in Arikan et al. [2003], we have a hierarchical solution, which derives fine-grained adjustments within the parameters of a coarse solution with a much smaller search space.

Second, the combinatorial structure of the dynamic programming problem can be exploited further. For example, in addition to the typical iterative algorithm, we have implemented a *factored* A* search for the optimum solution [Klein and Manning 2003]. A* search builds the dynamic programming table dynamically using a priority queue. Options are explored based not on the error accumulated so far but also using a heuristic estimate of the further cost that the option must incur. With factored A* search, we derive an accurate heuristic estimate by solving separate dynamic programming problems for sound and motion in advance.

3.3.2 Animation

Our strategy for realizing the animation is diagramed in Figure 7. The speech segments (in green) provide a rigid time-line. Each segment provides one or two fixed temporal landmarks. These correspond to the first and last prosodic peaks in the phrase. There is an additional landmark at the splice point between successive speech segments. We use these landmarks to describe two kinds of intervals in the final animation: *playback windows* and *blending windows*. Playback windows represent the frames of animation which draw only on a single segment of captured data. During playback windows, only temporal warping is applied. Our temporal warping method aligns the first frame of motion emphasis with the beginning of the window and aligns the final frame of motion emphasis with the end of the window. Intermediate frames are created by interpolation. We treat the interval between the prosodic peaks of each speech segment as a playback window. The initial and final segments of the utterance are also treated as playback windows, but the sound is extended with silence if necessary so that motions never need to be warped faster. In addition, if successive motions are selected to accompany successive speech segments, we bridge successive speech segments with a playback window.

Blending windows represent the frames of animation in which two segments of captured data are combined. We create blending windows for the whole interval without emphasis between successive speech segments, when we have to blend discontinuous motion data. We create the blend following [Witkin and Popović 1995]. We first find a smaller overlay window centered as close as possible to the join between speech segments. We create a warp target for the first clip of captured data at the end of the overlay window, by interpolating between the two motion clips. Before using the first segment, we interpolate it towards this target. During the blending window before the overlay window, just the interpolated frames from the first segment are used.

For the second clip we create an analogous warp target at the start of the overlay window. But the second animation is first translated to keep at least one foot close to stationary across the blend. After the overlay window, just the interpolated, translated frames from the second segment are used. During the overlay window, of course, the two (interpolated) motions are interpolated with each other.

4 Implementation and Results

Using this platform, we built an end-to-end generator for critique of play in the tutorial mode of a video game like Electronic Art's SSX snowboarding series. Our systematic design ensures that there is an utterance for every context across this scenario, that these utterances vary appropriately as the context changes, that they remain grammatical despite combining phrases in new ways, and that they index into available performances of speech and gesture. Figure 8 and the accompanying video shows synthesized examples of our character. We used our our skeleton to drive a model from Electronic Art's SSX 3 (Zoe), and rendered in Maya. In these examples, both speech and gesture are spliced together drawing on multiple utterances. Our character retains the fluid meaningful gestures of the original performance. The transformations required to stitching motions together and realign motion into synchrony with speech are unobtrusive, and the match between gesture and speech preserves much of the meaning and personality of the actor's original portrayal.

The data collection and analysis required to create this character was modest. Kate Brehm acted our script at the NYU motion capture lab. We captured her performance with a 12-camera Vicon motion capture system, a head-mounted microphone and a DAT recorder, and a digital video camcorder. The script involved 22 utterances and 102 distinct phrases; each complete performance took five minutes and yielded about two minutes of speech and motion.

Our prototype tools require a mix of text entry and *ad hoc* scripting. In a more established pipeline with direct, user-friendly interfaces (e.g., in an industrial game development setting), we expect that an added day or two in a motion capture studio and an added person-month of data preparation would suffice to put together 10 scenarios like the one we developed, each one designed so as to add 1-2 minutes of conversational animation to perhaps 10 minutes of other play (for a role-playing game). This is enough to support players in a two-hour experience in a seemingly unbounded world populated with individual talking characters who respond specifically to them. Achieving comparable variability with generic techniques for language processing and procedural animation would require a much greater initial investment in computational infrastructure.

The script and data for our prototype may be small, but our character's behavior is quite variable. Its grammar distinguishes 98 different states in game play, with two or more alternative wordings for each phrase in each state; the grammar generates more than 6000 phrase combinations. In the data-driven realization of these utterances, we can trade off optimization to achieve further variability. Note that the two are inherently at odds. Generalizing [Bulyko and Ostendorf 2002], we could optimizate simutaneously over content, speech and motion, but we would get just one, single lowest-cost realization per context: 98 utterances total.

To understand this tradeoff, we compared three solution methods for utterance construction: dynamic programming (DP); sampling from low-cost utterances (SS) by perturbing DP with a small random change to the score terms; or uniformly random realization (RR). We collected ten outputs for each of the 32 kinds of errors (320 examples per condition, covering a large fraction of the possible contexts). At one extreme, RR used all 59 available sound and motion clips and never repeated a selection of sounds or of motions (this means there were 320 different patterns of motions and also of sounds), but the median score of its utterances was 1300. (Good scores are typically in the range 10 to 50.) By contrast DP used 43 of the motion units and 52 sound units, and selected just 39 patterns of motion stretched as necessary to match up with 286 patterns

"That was ugly, dude.
On this run you forgot to set up your jump."

"Yikes, dude!
You nailed it"

Figure 8: Additional synthesized utterances.

of speech (which was more variable because it was more directly driven by content). The median score was now 16, and about one third of the cases involved a choice of speech units that was not optimal except in the context of the accompanying motions. Finally, SS used 53 motion units and 57 sound clips, and saw 159 different motion patterns and 309 distinct sound patterns. SS continued to offer engaging realizations, achieving a median score of 26; it seems that our approach can make highly variable use of available data for extended interaction with a character. At the same time, the DP results suggest that our approach can also work with small samples of selected data for high-quality lightweight realization which would be suitable for infrequent interactions. By analyzing the DP results, we were quickly able to construct a smaller collection of 18 frequent motion clips that DP could align with speech (across the 32 contexts) with a median score of 19.

Our implementation is designed to demonstrate the feasibility of the approach; a range of known techniques could improve the output animation, such as dynamic spatiotemporal constraints (e.g., to avoid foot skate) [Kovar et al. 2002b], spline warping to improve motion continuity [Kovar and Gleicher 2003], or search over frames of animation to find the best blending point for selected motions [Arikan et al. 2003]. We could also tune weights, ideally by perceptual metrics and machine learning, as pursued in speech synthesis research such as [Hunt and Black 1996] and under development for computer animation, as in [Reitsma and Pollard 2003]. Our manipulation of output speech is also rudimentary—at run-time, we derive a pitch error by comparing the pitch range of each segment to that seen originally after its predecessor; we distribute error evenly across the sentence by adjusting the pitch of speech samples up or down using the Praat speech analysis program (www.praat.org).

Other limitations reflect genuine trade-offs among alternative designs for interactive conversational characters. For example, displaying a range of affect with performance-based animation requires many consistent performances. Procedural animation can achieve such variability more systematically [Chi et al. 2000]. Conversely, handling open-ended vocabulary (such as names and numbers) requires more general speech synthesis with smaller units over a finer time-scale, and combining limited domain and general speech synthesis is a perennial challenge [Black and Lenzo 2000].

5 Conclusion

This paper has presented techniques for animating conversational characters from human performance. The central idea is to reason over natural units of communicative performance, and to retain the temporal synchrony and communicative coordination that characterizes peoples' spontaneous delivery. We work with a compact, systematic description of application utterances and automatically convert between multiple representations of this description to support the entire life-cycle of character development. This framework offers an explicitly multimodal architecture, in which every stage of design, analysis and computation works with coordinated and synchronized multimodal presentations.

For future work, we plan a comprehensive assessment of users' reactions to our animations, in tandem with building an ECA for a meaningful task and carefully controlling against any design flaws that remain in the new interface. We also hope to combine our results with other approaches to conversational animation. For example, given suitable databases of performance, our techniques could also be used to realize utterance specifications derived by rule-based planning methods, such as [Cassell et al. 1994], or derived by inference from text input, as in [Cassell et al. 2001].

Capturing detail of movement in the face and hands remains difficult, and the analysis we do of the data we have is intensive. The better and more reliably motion can be captured and analyzed automatically, the less expensive these techniques will be to use. The techniques required are the subject of ongoing research. But we have demonstrated in this paper that scientific principles and good design already suffice to create engaging, high-quality delivery interactive characters with reasonable effort. The possibility of streamlining the process only adds to the attraction of the approach.

Acknowledgments

This research ws supported in part by NSF (HLC 0308121) at Rutgers and ONR at NYU. Thanks to Kate Brehm for her wonderful realization of our character, to Loren Runcie and Jared Silver for help with animation, and to Electronic Arts for the use of Zoe from SSX 3. The presentation greatly benefits from comments by the SIGGRAPH reviewers.

References

ARIKAN, O., AND FORSYTH, D. A. 2002. Interactive motion generation from examples. *ACM Trans. Graph. 21*, 3, 483–490.

ARIKAN, O., FORSYTH, D. A., AND O'BRIEN, J. F. 2003. Motion synthesis from annotations. *ACM Trans. Graph. 22*, 3, 402–408.

BANGALORE, S., AND RAMBOW, O. 2000. Exploiting a probabilistic hierarchical model for generation. In *Proceedings of COLING*, 42–48.

BAVELAS, J. B., AND CHOVIL, N. 2000. Visible acts of meaning: An integrated message model of language in face-to-face dialogue. *Journal of Language and Social Psychology 19*, 2, 163–194.

BESKOW, J., EDLUND, J., AND NORDSTRAND, M. 2002. Specification and realisation of multimodal output in dialogue systems. In *Proceedings of ICSLP*, 181–184.

BESKOW, J. 2003. *Talking Heads: Models and Applications for Multimodal Speech Synthesis*. PhD thesis, KTH Stockholm.

BEUTNAGEL, M., CONKIE, A., SCHROETER, J., STYLIANOU, Y., AND SYRDAL, A. 1999. The AT&T Next-Gen TTS system. In *Joint Meeting of ASA, EAA, and DAGA*, 18–24.

BICKMORE, T. W. 2003. *Relational Agents: Effecting Change through Human-Computer Relationships*. PhD thesis, MIT.

BLACK, A. W., AND LENZO, K. A. 2000. Limited domain synthesis. In *Proceedings of ICSLP*, vol. II, 411–414.

BRAND, M. 1999. Voice puppetry. In *SIGGRAPH*, 21–28.

BULYKO, I., AND OSTENDORF, M. 2002. Efficient integrated response generation from multiple targets using weighted finite state transducers. *Computer Speech and Language 16*, 533–550.

CASSELL, J., PELACHAUD, C., BADLER, N., STEEDMAN, M., ACHORN, B., BECKET, T., DOUVILLE, B., PREVOST, S., AND STONE, M. 1994. Animated conversation: Rule-based generation of facial expression, gesture and spoken intonation for multiple conversational agents. In *SIGGRAPH*, 413–420.

CASSELL, J., SULLIVAN, J., PREVOST, S., AND CHURCHILL, E., Eds. 2000. *Embodied Conversational Agents*. MIT.

CASSELL, J., VILHJÁLMSSON, H., AND BICKMORE, T. 2001. BEAT: the behavioral expression animation toolkit. In *SIGGRAPH*, 477–486.

CASSELL, J. 2000. Embodied conversational interface agents. *Communications of the ACM 43*, 4, 70–78.

CERRATO, L., AND SKHIRI, M. 2003. A method for the analysis and measurement of communicative head movements in human dialogues. In *Proceedings of AVSP*, 251–256.

CHI, D., COSTA, M., ZHAO, L., AND BADLER, N. 2000. The EMOTE model for effort and shape. In *SIGGRAPH*, 173–182.

DECARLO, D., REVILLA, C., STONE, M., AND VENDITTI, J. 2002. Making discourse visible: coding and animating conversational facial displays. In *Proceedings of Computer Animation*, 11–16.

EKMAN, P. 1979. About brows: Emotional and conversational signals. In *Human Ethology: Claims and Limits of a New Discipline: Contributions to the Colloquium*, M. von Cranach, K. Foppa, W. Lepenies, and D. Ploog, Eds. Cambridge University Press, Cambridge, 169–202.

ENGLE, R. A. 2000. *Toward a Theory of Multimodal Communication: Combining Speech, Gestures, Diagrams and Demonstrations in Instructional Explanations*. PhD thesis, Stanford University.

EZZAT, T., GEIGER, G., AND POGGIO, T. 2002. Trainable videorealistic speech animation. *ACM Trans. Graph. 21*, 3, 388–398.

GLEICHER, M. 1998. Retargeting motion to new characters. In *SIGGRAPH*, 33–42.

HOPCROFT, J. E., MOTWANI, R., AND ULLMAN, J. D. 2000. *Introduction to automata theory, languages and computation*, second ed. Addison-Wesley.

HUNT, A. J., AND BLACK, A. W. 1996. Unit selection in a concatenative speech synthesis system using a large speech database. In *Proceedings of ICASSP*, vol. I, 373–376.

JURAFSKY, D., AND MARTIN, J. H. 2000. *Speech and Language Processing: An introduction to natural language processing, computational linguistics and speech recognition*. Prentice-Hall.

KIM, T., PARK, S. I., AND SHIN, S. Y. 2003. Rhythmic-motion synthesis based on motion-beat analysis. *ACM Trans. Graph. 22*, 3, 392–401.

KLEIN, D., AND MANNING, C. D. 2003. Factored A* search for models over sequences and trees. In *Proceedings of IJCAI*.

KOPP, S., AND WACHSMUTH, I. 2004. Synthesizing multimodal utterances for conversational agents. *Computer Animation and Virtual Worlds 15*, 1, 39–52.

KOVAR, L., AND GLEICHER, M. 2003. Flexible automatic motion blending with registration curvers. In *Symposium on Computer Animation*.

KOVAR, L., GLEICHER, M., AND PIGHIN, F. 2002. Motion graphs. *ACM Trans. Graph. 21*, 3, 473–482.

KOVAR, L., SCHREINER, J., AND GLEICHER, M. 2002. Footskate cleanup for motion capture editing. In *Symposium on Computer Animation*.

KRAHMER, E., RUTTKAY, Z., SWERTS, M., AND WESSELINK, W. 2002. Audiovisual cues to prominence. In *Proceedings of ICSLP*.

LANGKILDE, I. 2000. Forest-based statistical sentence generation. In *Applied Natural Language Processing Conference*, 170–177.

LEE, J., CHAI, J., REITSMA, P. S. A., HODGINS, J. K., AND POLLARD, N. S. 2002. Interactive control of avatars animated with human motion data. *ACM Trans. Graph. 21*, 3, 491–500.

LEE, S. P., BADLER, J. B., AND BADLER, N. I. 2002. Eyes alive. *ACM Trans. Graph. 21*, 3, 637–644.

LI, Y., WANG, T., AND SHUM, H.-Y. 2002. Motion texture: a two-level statistical model for character motion synthesis. *ACM Trans. Graph. 21*, 3, 465–472.

MCNEILL, D., QUEK, F., MCCULLOUGH, K.-E., DUNCAN, S., FURUYAMA, N., BRYLL, R., MA, X.-F., AND ANSARI, R. 2001. Catchments, prosody and discourse. *Gesture 1*, 1, 9–33.

MCNEILL, D. 1992. *Hand and Mind: What Gestures Reveal about Thought*. University of Chicago Press, Chicago.

PAN, S., AND WANG, W. 2002. Designing a speech corpus for instance-based spoken language generation. In *Proceedings of Int. Conf. on Natural Language Generation*, 49–56.

PARK, S. I., SHIN, H. J., AND SHIN, S. Y. 2002. On-line locomotion generation based on motion blending. *Proc. ACM SIGGRAPH Symposium on Computer Animation*, 105–111.

PELACHAUD, C., BADLER, N., AND STEEDMAN, M. 1996. Generating facial expressions for speech. *Cognitive Science 20*, 1, 1–46.

PERLIN, K., AND GOLDBERG, A. 1996. Improv: a system for interactive actors in virtual worlds. In *SIGGRAPH*, 205–216.

PIERREHUMBERT, J., AND HIRSCHBERG, J. 1990. The meaning of intonational contours in the interpretation of discourse. In *Intentions in Communication*, P. Cohen, J. Morgan, and M. Pollack, Eds. MIT Press, Cambridge MA, 271–311.

POPOVIĆ, Z., AND WITKIN, A. 1999. Physically based motion transformation. In *SIGGRAPH*, 11–20.

PULLEN, K., AND BREGLER, C. 2002. Motion capture assisted animation: texturing and synthesis. *ACM Trans. Graph. 21*, 3, 501–508.

REITER, E., AND DALE, R. 2000. *Building Natural Language Generation Systems*. Cambridge University Press.

REITSMA, P. S. A., AND POLLARD, N. S. 2003. Perceptual metrics for character animation: sensitivity to errors in ballistic motion. *ACM Trans. Graph. 22*, 3, 537–542.

ROSE, C., COHEN, M., AND BODENHEIMER, B. 1998. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics and Applications 18*, 5, 32–40.

SCHROETER, J., OSTERMANN, J., GRAF, H. P., BEUTNAGEL, M., COSATTO, E., SYRDAL, A., AND CONKIE, A. 2000. Multimodal speech synthesis. In *IEEE International Conference on Multimedia and Expo*, vol. I, 571–574.

SENEFF, S. 2002. Response planning and generation in the Mercury flight reservation system. *Computer Speech and Language 16*, 282–312.

SILVERMAN, K. E. A., BECKMAN, M., PITRELLI, J. F., OSTENDORF, M., WIGHTMAN, C., PRICE, P., AND PIERREHUMBERT, J. 1992. ToBI: a standard for labeling English prosody. In *Proceedings of ICSLP*, 867–870.

STEEDMAN, M. 2000. Information structure and the syntax-phonology interface. *Linguistic Inquiry 31*, 4, 649–689.

STONE, M., AND DECARLO, D. 2003. Crafting the illusion of meaning: Template-based generation of embodied conversational behavior. In *Proceedings of Computer Animation and Social Agents*, 11–16.

THEUNE, M., AND KLABBERS, E. 2001. From data to speech: A general approach. *Natural Language Engineering 7*, 1, 47–86.

WILLIAMS, L. 1990. Performance-driven facial animation. In *SIGGRAPH*, 235–242.

WITKIN, A., AND POPOVIĆ, Z. 1995. Motion warping. In *SIGGRAPH*, 105–108.