Gesture Modeling and Animation by Imitation

Michael Neff, Michael Kipp, Irene Albrecht, Hans-Peter Seidel

MPI–I–2006–4-008 September 2006

Authors' Addresses

Michael Neff
Max-Planck-Institut für Informatik
Stuhlsatzenhausweg 85
66123 Saarbrücken
Germany

Michael Kipp
Deutsches Forschungszentrum für Künstliche Intelligenz
Stuhlsatzenhausweg 3
66123 Saarbrücken
Germany

Irene Albrecht
Max-Planck-Institut für Informatik
Stuhlsatzenhausweg 85
66123 Saarbrücken
Germany

Hans-Peter Seidel
Max-Planck-Institut für Informatik
Stuhlsatzenhausweg 85
66123 Saarbrücken
Germany

Acknowledgements

First and foremost, the authors would like to thank Jay Leno and Marcel Reich-Ranicky for providing such excellent samples of gesturing as part of their normal routine. The authors are also grateful to Jürgen Trouvain for the phoneme transcriptions, Ari Shapiro for support with version 3 of the DANCE framework and all the participants of the evaluation experiment.

Abstract

Animated characters that move and gesticulate appropriately with spoken text are useful in a wide range of applications. Unfortunately, they are very difficult to generate, even more so when a unique, individual movement style is required. We present a system that is capable of producing full-body gesture animation for given input text in the style of a particular performer. Our process starts with video of a performer whose gesturing style we wish to animate. A tool-assisted annotation process is first performed on the video, from which a statistical model of the person's particular gesturing style is built. Using this model and tagged input text, our generation algorithm creates a gesture script appropriate for the given text. As opposed to isolated singleton gestures, our gesture script specifies a stream of continuous gestures coordinated with speech. This script is passed to an animation system, which enhances the gesture description with more detail and prepares a refined description of the motion. An animation subengine can then generate either kinematic or physically simulated motion based on this description. The system is capable of creating animation that replicates a particular performance in the video corpus, generating new animation for the spoken text that is consistent with the given performer's style and creating performances of a given text sample in the style of different performers.

Keywords

human modeling, gesture, character animation

Contents

1 Introduction

People are engaged by characters with interesting personalities. However, creating quality animation for generic characters that correctly coordinates appropriate gestures with spoken text is already a challenging task. Generating movement that reflects a particular personality significantly increases the challenge of the gesture animation task, yet it is a goal towards which we must strive. This work describes one approach towards that goal. We present a system [1] that allows the gesturing pattern of specific individuals to be modelled and then generates animation for new text from this model, complete with appropriate gestures and body movement that reflect the original subject.

The system operates in two phases: a pre-processing phase and a fully automatic generation phase. The preprocessing stage is shown in Figure 1.1. Producing animation of a particular individual begins by collecting a video corpus for that person. In our case, we used two talk show hosts as subjects, employing less than ten minutes of film of each. During an analysis step, the video is hand annotated in the tool ANVIL [19], and both gesture and more general animation data are extracted. A statistical model called a *Gesture Profile* is built based on the annotated data. In addition, an *Animation Lexicon* is constructed that contains data such as

[1] This work is also published as technical report [34].

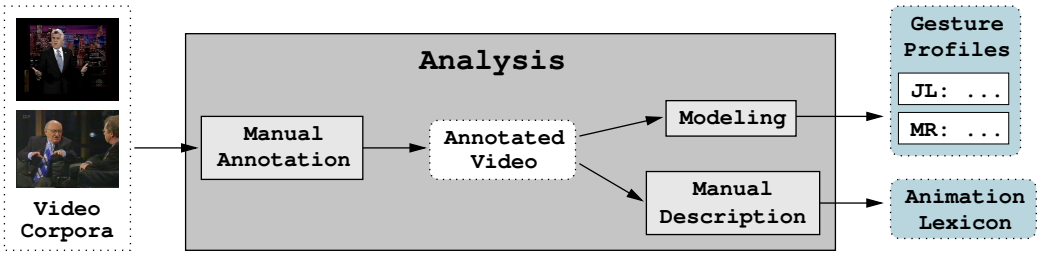

Figure 1.1: Preprocessing phase of the system

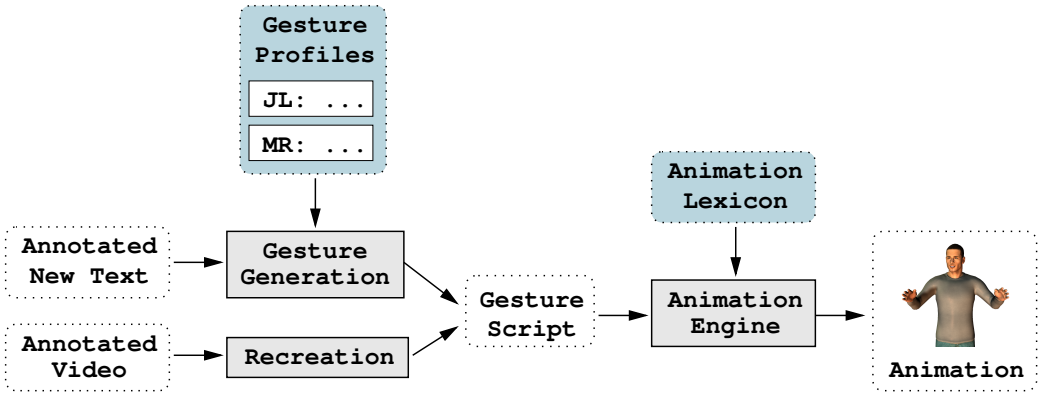

Figure 1.2: Automatic generation process.

the normal hand orientation for each gesture that we model. These two components provide the input for the gesture generation and animation stage. This tool supported analysis step allows us to generate a particularly wide range of gestures.

Once the initial analysis phase is complete, the generation process is fully automatic, as summarized in Figure 1.2. There are two paths in this pipeline. The bottom path shows *re-creation*. Here, the annotated data from the video corpus is mapped directly to a *gesture script*, which is then animated. Recreation can be used to produce animations of any annotated segment in the corpus. This is useful for validating the annotation and creating animation of a specific performance. The second path in the pipeline generates animation for any *novel* tagged text, which need not be in the video corpus. The gesture script is generated from the statistical model for the individual specified by the user. This gesture script is passed to the *Animation Engine*, which further refines the description of the motion, using data from the animation lexicon and a set of rules described below. The animation engine produces final animated output either kinematically, or using dynamic simulation, at the user's option.

Traditionally, modeling gesture production and gesture animation are handled by well separated systems. For instance, the gesture specification system might only produce a gesture name, e.g. "beat", that is then rendered by the animation system by playing a pre-existing clip. In this work, we take a more tightly coupled approach that raises interesting issues of data representation and flow: What data gets generated where? Some of the detailed information needed to animate a gesture is best stored as part of the gesture model, i.e. on the generation side. This information can be related to the model of a particular performer, the correlations within a stream of generated gestures, synchronization with speech, or be definitional information for a particular gesture. Modeling this data as part of

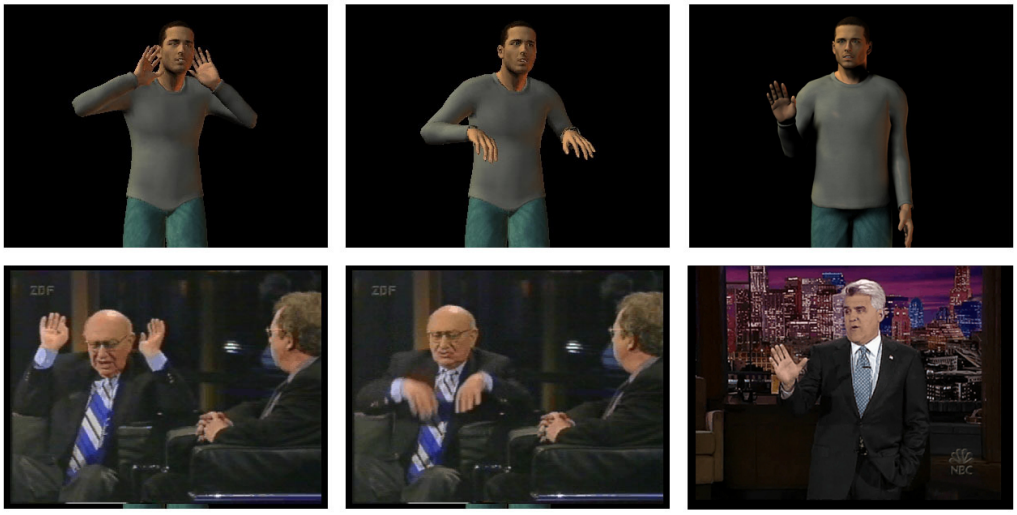

Figure 1.3: Subjects JL and MR with significant gesture style differences were analyzed and imitated in kinematic and dynamic animations.

the generation process allows greater control for the gesture generation system and reduces the burden of how much the animation system needs to "know" and model. It frees the animation system from needing to access the gesture model. At the same time, representing data on the generation side can be costly in terms of modeling effort as it requires sufficient video footage, annotation work and the construction of statistical or rule based models. In keeping with principles of data encapsulation, we also wish to minimize what the gesture system needs to know about the animation system. To negotiate this trade off, our gesture generation system produces sparse data that captures the key definitional aspects of a gesture and provides good control. The details of the motion are filled out by the animation system, which also has additional information about the figure being modeled, gesture types and the controls available. Specifically, the animation system will complete timing information, deal with spatial conflicts and add in a more rich description of gesture form as it augments the sparse data it receives from the generation stage.

In comparing our system to a motion capture approach, as well as noting the greater control afforded the generation engine in our approach, it is worth examining the range of gestures our system can produce. We currently model 28 different gestures in our animation lexicon, each of which can be generated with either or both hands in any of hundreds of spatial locations and with an arbitrary number of after-strokes (Chapter 3). Thousands of different combinations are possible and the animation lexicon can be easily extended. Developing an appropriate motion

database to cover this space would be a daunting, if not prohibitive, task. Another strength of our technique is that it can be used on any subjects for whom there is adequate film of them gesturing. This means that it can potentially be used on subjects that are no longer alive or who cannot otherwise be motion captured due to cost or availability.

Two talk shows hosts, JL (Jay Leno) and MR (Marcel Reich-Ranicki), are used as subjects in this paper. They have different gesturing styles (see Figure 1.3) and also speak different languages. This illustrates an important point: our technique can be used to create gestural animation for text in languages different to that spoken by the subject. In our animation system, we employ a skeleton model containing 89 degrees of freedom, including six degrees for world space orientation and location and 21 degrees of freedom for each hand.

In summary, this system offers the following contributions:

1. The production of high quality human gestural and facial animation, synchronized with speech.
2. The recreation of specific gestural performances from input video.
3. The creation of new performances in a style consistent with a given subject, but with novel text.
4. The use of physical simulation and tension control to improve the quality of gestural animation.

2 Background

To generate and animate gestures from an input of tagged text is a fairly recent endeavor, pursued in an interdisciplinary arena requiring competencies from computer animation, artificial intelligence and psychology. Cassell *et al.* [4] developed a rule-based system that generates audiovisual speech, intonation, facial expression, and gesture by working on the input text's information structure which is still common practice today. Another common practice is to synchronize the gesture stroke to the accented syllable of the coexpressive word, although, as we will show, it makes sense to sometimes synchronize the stroke with a different part of the sentence. Using the same agent as Cassell *et al.*, Noma *et al.* [36] built the Virtual Presenter where gestures can be added to a text manually or with keyword-triggered rules. Animated gestures are synchronized with the following word. While the number of possible gestures is very small the focus was on how to implement meaningful rules from the literature on good public speaking. The system takes into account posture and eye contact with the audience. A more complex generation system is the Behavior Expression Animation Toolkit (BEAT) [5]. It takes plain text as input and first runs a linguistic analysis on it before generating intonation, facial animation, and gestures. Gestures are generated using hand-made rules and are selected using priority values. While our system shares the overall goal of BEAT, to create accompanying gestures for a given text, the are a number of differences. In BEAT, a gesture is basically a "black box" that is triggered by a hand-made rule. In contrast, our system triggers gestures probabilistically and plans both the gestures' internal structure (phases, timing, shape) and macro-structure (by creating so-called *gesture units* [18]). Moreover, we produce a wide range of 28 different gestures[1] while BEAT seems to focus on very few samples. Most importantly, our approach produces not only natural-looking animations but a *character-specific* gesture style that intends to capture the individual differences of human beings.

[1] This does not even take into account the large variation that we achieve with different positions and varying phase structure, especially multiple strokes.

Stone *et al.* [44] re-create a specific person's gesturing style by re-arranging pre-recorded chunks of audio and motion captured pieces of full-body movement. Possible sentences are defined by a simple grammar. The corresponding utterance is assembled from those speech phrases and gestures that match the communicative function and minimize the required amount of time warping and blending. Other relevant animation systems include EMOTE [6] which presents a kinematic system for expressive variation of arm and torso movements that is based on the analysis of Laban. Hartmann et al. [13] present a kinematic animation system that realizes a gesture language that they have developed. Our gesture representation shares many features in common with theirs. We extend their representation in several ways: an additional spatial dimension is modeled (swivel angle), both world space and local hand orientation constraints are supported, additional movement features like posture, trajectory and tension changes are modeled, and a more complex representation for after-strokes is developed. Hartmann *et al.* [14] extend their system to add expressivity to their gesture synthesis by varying activation, spatial and temporal extent, fluidity and repetition. [37] is also concerned with gesturing style. A style consists of a dictionary of meaning-to-gesture mappings, motion characteristics, and modality preferences. Combining style dictionaries yields mappings for new cultural groups or individuals. In contrast to our approach, their styles are hand crafted and model the behaviour of stereotypic groups instead of real individuals. In addition, the placement and frequency of gestures is fully determined by tags in the input text, and gestures are modeled at a comparatively coarse level since the paper's focus is a style description language, and it is not concerned with animation issues.

Kopp *et al.* [23, 24] present a gesture animation system that makes use of neurophysiological research and generates iconic gestures from object descriptions and site plans when talking about spatial domains, e.g. giving directions. Iconic gestures resemble some semantic feature of an object referred to in the co-occurring speech. In contrast, in our approach the domains are mostly non-spatial. Many iconics that occur in everyday conversation are either metaphoric and therefore standardized [28] or verging on the emblematic (e.g. gestures for actions like drinking or counting) and thus also standardized. Therefore, Kopp *et al.*'s approach and ours can be considered complementary. More in line with our approach is de Ruiter's [8] Sketch Model where both gesture and speech originate in the same module called a *conceptualizer*. Gestures are processed in data structures with unbound variables, so-called *sketches*, that can be filled according to context and using a gestuary of concept-to-shape entries. A *gesture planner* fills the remaining parameters like body part (which hand(s)) and spatial locations and builds a final motor program for the articulators. The model is not implemented but can predict certain phenomena in gesture-speech synchronization. Our ap-

proach shares the processing of underspecified gesture structures which we call gesture frames.

Neff and Fiume [32] present a system for modelling gesture-like movements using physical simulation, but do not model a complicated range of gestures or combine them with speech. Physical simulation has been used for many years to generate character motion with two main approaches emerging: optimization techniques that use physical laws as constraints [47, 39] and simulation techniques that forward simulate Newton's laws to generate motion [15]. We take a simulation approach, and in particular, follow on work in hand-tuned control [15, 9] where a proportional derivative (PD) controller is used at each character Degree of Freedom (DOF) to generate the required torques to make it move. As we have an underlying kinematic motion representation, our approach is also similar to the use of physical control to track motion capture data presented by [49], and our hand model is similar to [38]. To our knowledge, this is the first use of forward simulation on a character of this complexity that must synchronize its movements with speech.

3 Understanding Gesture

A good way to approach a concept as diffuse and organic as gestures is to look at their temporal structure which can be nicely described in terms of phases, phrases and units [28, 22, 18]. A single *gesture* can be described as consisting of a number of consecutive movement phases. This can be expressed by the following rule[1]:

$$\text{GESTURE} \rightarrow [\ \textbf{preparation}\ ]\ [\ \textbf{hold}\ ]\ \text{STROKE}\ [\ \textbf{hold}\ ] \tag{3.1}$$

Only the stroke phase must occur in every gesture, all other phases are optional. The stroke is the "most energetic" and "meaning-carrying" phase of the gesture while in the preparation phase the hands are moved to the stroke's start position. The stroke can consist of multiple repeated movements which would make it a *multi-stroke*[2]. Since the first stroke in a multi-stroke is often the most pronounced and the following strokes have similar form but look weaker than the first, we call the first stroke the *main stroke* and all subsequent strokes *after-strokes*:

$$\text{STROKE} \rightarrow \textbf{main\_stroke}\ (\textbf{after\_stroke})^{*} \tag{3.2}$$

The *hold* phases in rule (3.1), before and after the stroke, are optional pauses, usually interpreted as a means to correctly synchronize the stroke with accompanying speech. A complete GESTURE is also called a gesture phrase (g-phrase) in the literature. Opposing McNeill's claim that every gesture has a stroke, Kita *et al.* [22] found that some gestures have a single meaningful still phase instead, called an independent hold. We therefore distinguish two principal gesture types, stroke gestures (S-GESTURE) and hold gestures (H-GESTURE), and expand rule (3.1) to the following three rules:

[1] Nonterminals are set in smallcaps, terminals in boldface, and optional elements are put in square brackets.

[2] Kita *et al.* calls them *multiple strokes*, Hartmann *et al.* [12] calls them *repeats*.

$$\text{GESTURE} \rightarrow \{\ \text{S-GESTURE} \mid \text{H-GESTURE}\ \} \tag{3.3}$$

$$\text{S-GESTURE} \rightarrow [\ \textbf{preparation}\ ]\ [\ \textbf{hold}\ ]\ \text{STROKE}\ [\ \textbf{hold}\ ] \tag{3.4}$$

$$\text{H-GESTURE} \rightarrow [\ \textbf{preparation}\ ]\ \textbf{hold} \tag{3.5}$$

We call a *rest position* a pose where the hands either hang down at the side or are supported in some way: e.g., arms lie on an arm rest, arms are folded, hands are in pockets or are locked behind one's back. A gestural excursion always starts from a rest position, can encompass one or more gestures and finally returns to a rest position. Such an excursion is called a *gesture unit* (g-unit). For gesture generation, the g-unit is an important organizational entity as it groups together multiple gestures in one continuous flow of movement. A unit always ends with a *retraction* movement to a rest position.

$$\text{UNIT} \rightarrow (\text{GESTURE})^{+}\ \textbf{retraction} \tag{3.6}$$

A unit can consist of a single gesture. McNeill [28] actually found that his subjects frequently perform only one gesture per unit (only 44% of the time would his subjects perform more than one gesture per unit). However, his subjects consisted of people who were neither trained nor experienced in speaking in public or on TV. In contrast, our data of professional TV performers shows a completely different picture. Table 3.1 shows how often the different g-unit sizes occured. Our speakers frequently combine multiple gestures to units: MR uses units with more than one gesture 66.7% of the time, JL 64.3% of the time. We believe that this is one reason why JL's and MR's gestures are enjoyable to watch: the speakers produce a fluent stream of continuous gestures instead of isolated singleton gestures. One aim of our project was to transfer this quality to synthetic agents.

3.1 Gesture Lexicon and Lexemes

It follows from the encountered usage patterns that since we want to produce gesture behaviour that looks characteristic for a certain person, we need to produce a *broad* spectrum of gestures. Previous work focussed on a limited range of specific gestures in order to work out details, e.g., about the semantics-form relationship between speech and iconic gesture [23]. However, in our target domain, iconic gestures that need a deep understanding of semantics and form rarely occur.

	1	2	3	4	5	6	>6	total number of units
JL	35.7 %	15.7 %	17.1 %	5.7 %	11.4 %	5.7 %	8.6 %	70
MR	33.3 %	16.7 %	11.1 %	14.8 %	9.3 %	3.7 %	11.1 %	54
McNeill	56 %	14 %	8 %	8 %	4 %	2 %	8 %	254

Table 3.1: Number of gestures per unit for our speakers JL and MR in comparison with McNeill's subjects. The table shows that JL and MR use units with more than one gesture much more often than McNeill's subjects.

In everyday conversations, but also in talk shows and formal presentations, human speakers use mostly gestures where no strong semantic function is visible. McNeill calls these gestures *metaphorics* since the relationship between the gesture and what is said is only etablished through an abstract metaphor. For instance, in a *progressive* gesture the speaker's hands revolve around each other in circles. McNeill argues that the gesture refers to the abstract notion of a forward rotating wheel which in turn refers to a co-occuring word in speech like, e.g., "going", "developing" or even "future". These are the gestures we focus on. However, do these gestures share a common form or is their shape totally arbitrary and invented on the fly?

While it is common knowledge that emblematic gestures (e.g. the victory sign or the thumbs-up gesture) are drawn from a shared, though culture-specific, lexicon, it became clear only recently that this is also true for more abstract gestures. [46] showed for a number of speakers that they use metaphoric gestures from a shared lexicon of forms (see also [20]). Although each speaker applies slight variations and only uses a subset of these gestures, there is basically one large reservoire of gestures that all speakers draw from. In our approach to gesture generation we exploit this insight to represent gestures as lexicon entries, so-called *lexemes*, which can be considered equivalence classes with respect to form and function.

Figure 3.1: Two occurrences of the lexeme *Wipe* in our corpus (left: speaker MR, right: speaker JL). In each example, the first frame shows the end of the prepatory movement and the second frame shows the end of the stroke.

Kipp [20] collected a lexicon of gestures for two German TV show hosts. To this work, we added a new speaker with a different language: the American talk show host JL. We assembled a gesture lexicon of 39 lexemes[3] and annotated a video corpus (Chapter 4). Of this gesture lexicon, MR uses a subset of 31 lexemes and JL uses 35. The large overlap of 27 lexemes that both MR and JL use supports the hypothesis of a shared lexicon of gestures that all people use. Figure 3.1 shows the lexeme *Wipe*, performed by MR (left) and JL (right).

[3]Note that the animation engine currently only models a subset of 28 lexemes. This is partly due to rare lexemes that occur in the video corpus but were never generated in our examples because of their low probability and partly due to gestures whose shape must be determined by

JL		MR	
lexeme	*%*	*lexeme*	*%*
Cup	24.4	Cup	6.9
PointingAbstract	8.9	RaisedIndexfinger	6.9
PointingPerson	6.7	FlingDown	5.8
HandClap	6.7	Wipe	5.8
Shrug	6.2	Beat	5.3
Progressive	4.4	Calm	5.3

Table 3.2: The table shows the six most frequently used lexemes for each speaker and how often the particular lexemes are used (in %).

While the gesture lexicon represents what is *shared* between speakers, the specific subset that each speaker uses and the frequency of each lexeme are significant aspects for modeling *interpersonal differences*. As Table 3.2 shows, the speakers differ significantly in what lexemes they use and how often they perform a particular lexeme. As we will show in the following section, we furthermore model the variations of gesture form between each speaker and generate particular lexemes in correlation with a speaker's tendency to use those lexemes for a given speech segment.

semantic knowledge not modeled by our system.

4 Analysis

To automatically generate and animate gestures in a speaker-specific style, a human speaker has first to be studied and analyzed. Using a combination of manual labour and automatic data extraction, the key factors of speaker gesture behaviour are then stored in machine-readable form for automatic gesture generation and animation.

Figure 4.1 gives a schematic overview of the analysis process. First, a video corpus for each speaker is annotated by hand. Both speech[1] and gestures are transcribed by human coders. This annotated corpus is used for three purposes. First, all annotated gestures are stored in the GestureDB database, as blueprints for automatic generation. Second, key properties for each gesture lexeme, especially the relationship between gesture and speech, are modeled with statistical tables and average values. Both the GestureDB and the statistical model are stored in speaker-specific *gesture profiles*. Third, for animation, speaker-specific lexeme properties are modeled in the *animation lexicon* (Section 4.1.3).

[1] Word and phoneme boundaries and timings are marked. At the minimum, phoneme information could be extracted automatically using current tools e.g. CMU Sphinx.

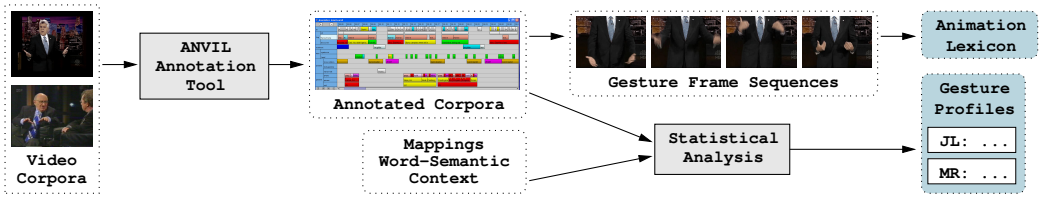

Figure 4.1: Analysis pipeline.

4.1 Subject Annotation

The video annotation serves to translate essential concepts in the source video material into a machine-readable form. A human coder annotates linguistic entities (e.g. words) and gestural entities, including temporal boundaries, on tracks on an annotation board (Figure 4.2). In order to check how well the annotation captures what happened in the original video we can feed it directly to the animation system, doing what we call a re-creation of the original behaviour (Figure 1.2). This is similar to what Frey [11] called re-animation. The manual annotation is a work-intensive process: 1 minute of video takes about 90 minutes of coding by a human coder. However, coding can be done by anyone after a brief period of training; no special knowledge of animation or linguistics is required. To support manual annotation, we use the video annotation tool ANVIL [19] and the phonetic analysis tool PRAAT [2].

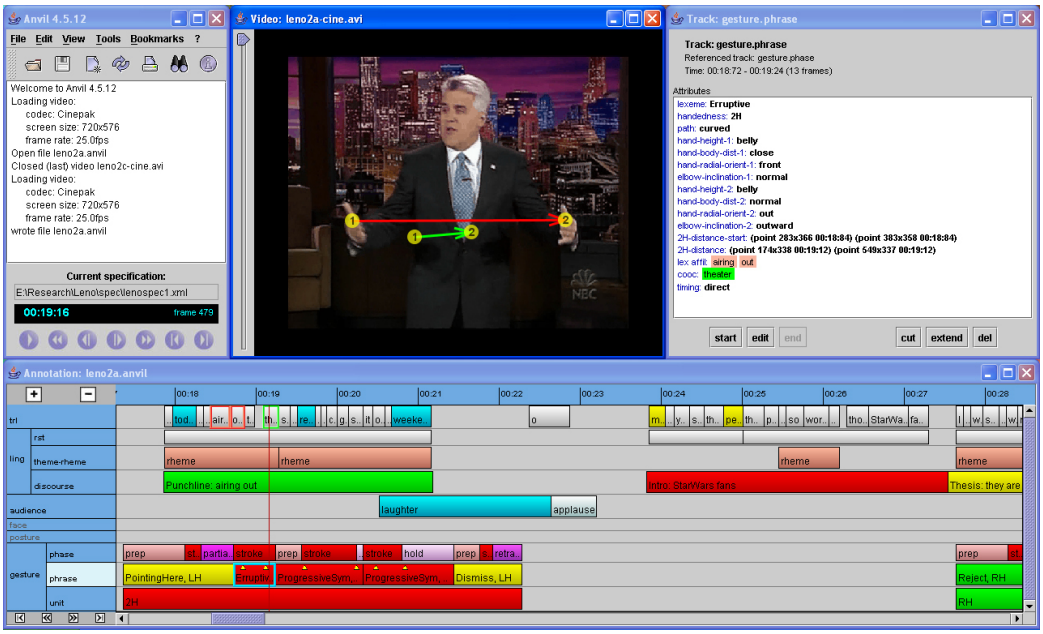

Figure 4.2: The ANVIL video annotation tool allows human coders to efficiently encode time-aligned information for digital video. In analysis, for each speaker a video corpus is transcribed for statistical modeling of gesture behaviour. The bottom window contains the multi-track annotation board where coding takes place.

4.1.1 Speech Annotation

The linguistic part of the annotation consists of coding words, discourse segments and information structure. We use the PRAAT tool to perform a word-by-word orthographic transcription of the utterance, including the words' boundaries, which is imported to ANVIL. Words must be grouped into senctence-like units. We use clauses as defined by Rhetorical Structure Theory (RST) [26]. However, any kind of discourse segmentation works with our approach. Finally, information structure is annotated using the concepts of theme, rheme and focus [43]. The theme is the part of the utterance that links the utterance to the previous discourse and specifies what the utterance is about, whereas the rheme relates to the theme and specifies something novel or interesting about it. Following Steedman [43] we also annotate the focus, which is the part of the rheme or theme that distinguishes the rheme/theme from other alternatives the context makes available. We make the simplifying assumption that the emphasized word or phrase is the rheme's focus. For example:

During the battle [rebel spies managed to steal secret plans to the Empire's ultimate weapon the Death Star]$_{rheme}$

In the example the first three words refer to a battle that is introduced in the preceding sentence which makes it the *theme* of the utterance. The bracketed part, the *rheme*, introduces the new information. And since "the Death Star" is emphasized it is the *focus* of the rheme.

4.1.2 Gesture Annotation

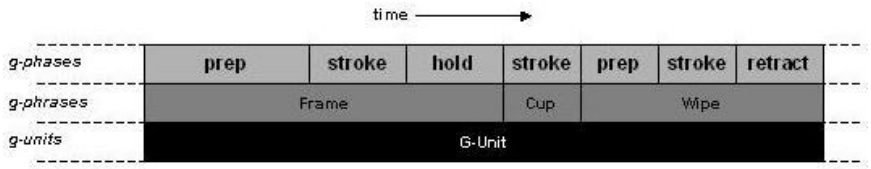

Figure 4.3: Gesture annotation entities on three tracks.

The gestural part of the annotation follows the hierarchical organization of gestures in phases, phrases and units, as described by rules (3.3)-(3.6) in Chapter 3. The human coder transcribes gestures in the video by adding *annotation elements* to three gesture annotation tracks in ANVIL (Figure 4.3). For each annotation element, the coder specifies begin and end times and then fills a number of attributes. In the top track, gesture phases are transcribed according to Kita *et al.* [22]. The

Height	*Distance*	*Radial Orientation*	*Elbow Inclination*
above head	far	far out	orthogonal
head	normal	out	out
shoulder	close	side	normal
chest	touch	front	touch
abdomen		inward	
belt			
below belt			

Table 4.1: Our three dimensions for hand position and one dimension for elbow inclination are divided into discrete intervals for annotation.

annotation elements contain one attribute for the phase type: preparation, stroke, hold etc. The coder has a second attribute to specify the number of strokes if the phase is a multi-stroke.

On the second track, several consecutive phases are combined into a gesture (e.g. Frame, Cup and Wipe in Figure 4.3). Following Kipp [20] we annotate the following attributes for each gesture: lexeme, handedness, lexcial affiliate and co-occurrence (Table 4.2). The lexeme denotes the lexicon entry that the gesture complies with (e.g., Frame, Cup, Wipe). Handedness denotes the executing hand(s). The lexical affiliate is the word or phrase that corresponds to the meaning or function of the gesture. For instance, "he" or "this" for a pointing gesture or "destroy" for a metaphoric progressive gesture. Since the lexical affiliate and the gesture do not always co-occur, the coder also specifies the word that the gesture co-occurs with.

We extended this scheme by adding information about the shape of the gesture. Since this data is expensive to annotate we devised a minimal coding scheme that is expressive enough to re-create the original gesture to a reasonable degree [21]. For each gesture, the coder specifies the trajectory (curved or straight) and hand/arm positions at the beginning and end of the stroke (s-gesture) or at the beginning of the independent hold (h-gesture).

Each hand/arm position is specified by a 4-vector $p = (h,d,r,s)$ for height, hand-body distance, radial zone and arm swivel angle. Each dimension of p has 5-7 discrete values $\{\text{above\_head}, \text{head}, \ldots, \text{below\_belt}\}$ (Table 4.1 and 4.2). For bihanded gestures, we additionally specify the hand separation (see Figure 4.2). For this, we extended ANVIL to allow graphical annotation where coders can edit 2D points on the video screen and store these points in an annotation element. Hand separation is annotated with two points, located at the middle of each palm, in the gesture phrase annotation element (middle layer in Figure 4.3). The shoulder width is also

encoded property	*encoding*	*description*
lexeme	{Cup, RaisedIndexfinger, Wipe, Progressive, ...}	Name of the gesture in the shared lexicon of conversational gestures.
handedness	{LH, RH, 2H}	Hand which performed the gesture (LH/RH), or both (2H).
lexical affiliate	*link to word(s) in the speech track*	The word or phrase in speech that semantically corresponds to the gesture, e.g. "you" for a deictic gesture to the addressee, or "going" for the gesture "Progressive".
co-occurrence	*link to word(s) in the speech track*	The words that temporally co-occur with the stroke of the gesture.
trajectory	{straight, curved}	Whether the gesture trajectory is straight or curved.
location 1	4-vector (height, hand-body distance, radial zone, arm swivel)	Location of the hand(s) at the beginning of the stroke or independent hold.
location 2	4-vector (height, hand-body distance, radial zone, arm swivel)	Location of the hand(s) at the end of the stroke. *Only for s-gestures.*
shoulder width	screen distance	Width of the shoulders in the current frame (used for normalizing hand separation). *Only for 2H gestures.*
hand separation 1	screen distance	Hand separation at the beginning of the stroke or independent hold (empty if angle to viewer is unsuitable). *Only for 2H gestures. Empty if view angle unsuitable.*
hand separation 2	screen distance	Hand separation at the end of the stroke (empty if h-gesture or angle to viewer is unsuitable). *Only for 2H s-gestures. Empty if view angle unsuitable.*

Table 4.2: The human coder specifies a number of properties for every gesture in the video. The g-phrase annotation element captures the gesture's semantic and temporal relation to speech and its form and development in space.

encoded and used to normalize hand separation. While there have been more precise approaches to transcribing positional features (e.g. [11] or [27]) our scheme was designed to be quick and economic to use while still incorporating sufficient information to re-create the original gestures.

On the third annotation track, the coder groups together contiguous gestures that are not interrupted by a full retraction to a single unit. Every unit thus ends with a full retraction unless the video ends in mid-gesture. The unit element also stores the retraction position of the unit's last gesture (e.g., hands at side or hands clasped).

speaker	*duration*	*#phases*	*#gestures*	*#units*
JL	9:04	574	229	70
MR	8:31	496	192	54

Table 4.3: The size of the annotated corpus for speakers JL and MR.

In Table 4.3 we show the size and contents of the annotated corpus for the two speakers JL and MR. Both corpora are of similar size. It is also interesting that both speakers seem to have a similar gesture frequency, since speakers can differ noticeably in that respect [20].

4.1.3 Producing an Animation Lexicon

The animation lexicon is created as part of the annotation process and contains additional data for each gesture lexeme. The ANVIL annotation tool automatically generates images of each lexeme that can be used as reference when defining the lexicon (Figure 4.1). For straight trajectory poses, these images show the start and end pose of the stroke. For curved trajectory poses, two internal frames are also generated. In our experience it takes on the order of a minute to a few minutes to annotate one gesture.

The animation lexicon contains three main types of data: hand orientation, torso posture and data for after-strokes. Hand orientation is specified by rotation around the forearm and two rotational degrees of freedom in the wrist. Either local values can be specified or constraints can be given in world space or chest space. Torso posture includes spine and collar bone movements that are either definitional for the gesture or characteristic of the particular character. We use a reduced DOF posture parameterization based on [33]. For many gestures, no torso information was defined, but it can be very important for certain gestures.

Recall that after-strokes are the small strokes following the main one in a multi-stroke. They generally carry similar meaning, but may differ in form and extent from the main stroke. They are generally smaller in amplitude and confined more to wrist and forearm movement. The prep and stroke data for these movements consists of forearm rotation, hand rotation, vertical or horizontal positional offsets, and elbow bend offsets. One prep and stroke are optionally specified for each after-stroke and they are then repeated for each repetition.

The lexicon also includes additional data that can be definitional for certain gestures such as warps to transition curves to change the timing profile and amplitude values for progressives.

Animation lexicons are character specific, but we found for our two characters that most of the data from one lexicon can be used directly in the second. Posture changes were the main data that was changed across subjects.

4.2 Building a Gesture Profile

The annotated corpus is used to build a profile for the speaker's gesture behaviour. The profile consists of the sample database, GestureDB, a statistical model and average values. For the GestureDB, the annotated information for each gesture in the corpus is stored as a reproducible "gesture sample" of the specific speaker. These samples can be seen as high-level movement patterns that can be easily modified in a meaningful way.

The statistical model is automatically computed from the annotations. It models estimated probabilities and is used in generation to trigger gestures, to predict where they are placed relative to speech, and to determine parameters like handedness and frequency. To build the model, the speech transcription needs a two-step preprocessing. In a first step, morphological analysis maps words to their lemma (e.g. striking→strike, won→win). In a second step, phrases, consisting of lemmas and/or words, are mapped to semantic tags, based on the assumption that similar gestural forms can express the meaning of the subsumed words. For example, the semantic tag INDETERMINATE replaces words like "somewhat" and "some". In our approach we employ look-up tables both for morphological analysis and semantic tagging. However, both tasks could be automatized using off-the-shelf software[2] or semi-automatic approaches[3]. Following Kipp [20] we use the semantic tags and the lexical affiliate annotations to estimate the conditional probability

[2]For instance, MORPHIX for lemmatization [10].

[3]For instance, using WordNet [29] for semantic modeling as used in BEAT [5].

of gesture lexeme l occurring with semantic tag s over our corpus C=(G, S) consisting of all occurring gestures G and semantic tags S by

$$\hat{P}(l|s) = \frac{\#\{g \in G : \text{lexeme}(g) = l \wedge \text{lexaffil}(g) = s\}}{\#S}$$

The probablities implicitly define a probabilistic concept-to-gesture mapping. Since the semantic tags are language-independent the resulting gesture profiles can be used for any target language; we use them for German and English. To model a speaker's preferred handedness and handedness shift patterns we utilize the unigram probability estimation $\hat{P}(h)$, the bigram estimation $\hat{P}(h_i|h_{i-1})$ and lexeme-relative handedness $\hat{P}(h|l)$, where $h \in \{LH, RH, 2H\}$.

Observation showed that the way multiple strokes (i.e. repeated strokes) are used or not used can be very characteristic for a speaker. We therefore store the average number of strokes $\mu_{strokes}$ per lexeme. To model the timing offset between gesture and speech we also record the average time difference ΔT_{end} between end of word and end of stroke (for hold gestures we record the start time difference ΔT_{start}). Finally, on a higher level we record gesture rate which is the number of gestures per minute because the amount of gesture activity also seems to be quite characteristic for a given speaker.

5 Gesture Generation

Once a speaker profile is created, our system can process any text input[1] and produce an animation with accompanying gestures. This "runtime" system consists of two components: the NOVA[2] gesture generator, decribed in this section, and the animation engine, described in Chapter 6. NOVA processes the input text and produces a gesture script for the animation engine (Figure 5.1).

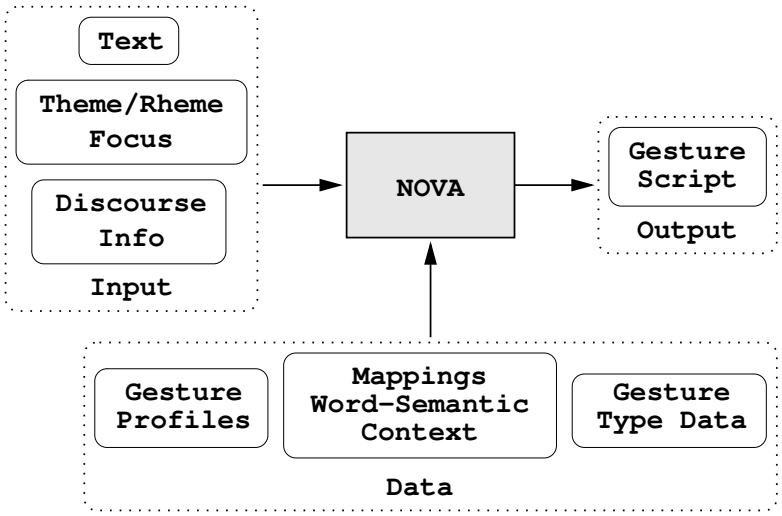

Figure 5.1: The NOVA gesture script generator.

The input text must be segmented into utterances and contain information about the theme, rheme and focus of each utterance (Section 4.1.1). For gesture generation, the input is transformed to a graph structure which is then processed in four stages:

[1] Although the input can be arbitrary, it must contain additional information as described later.

[2] **NO**n**V**erbal **A**ction Generator

1. Gesture creation
2. Gesture selection
3. Unit creation
4. Unit planning

The output is written to a gesture script which contains character-specific gestures, organized in units, with locational and timing parameters for animation.

5.1 Gesture Frames and Generation Graph

To generate, select and plan the gestures we use a graph structure to represent both speech and gestures (Figure 5.2). Generated gestures are inserted as arcs in the graph and represented as feature structures, so-called *gesture frames*. The underspecified frames become gradually enriched during generation using the speaker's profile and contextual constraints.

The input text is used to construct the initial graph. For each word, a node representing the begin time is created. While nodes represent time points, arcs represent concepts with a temporal duration: words, utterances, theme/rheme/focus. Words are mapped to lemmas that are added as arcs. Likewise, lemmas are mapped to semantic tags and added to the graph.

5.2 Gesture Generation Algorithm

5.2.1 Gesture Creation

In the first step we produce many candidate gestures for the given text by adding gesture arcs to the graph. For this, we use the concept-to-gesture mapping from the speaker's gesture profile (Section 4.2). For each rheme ρ, for each semantic tag s in ρ, we produce an underspecified gesture frame of lexeme l iff $\hat{P}(l|s) > 0.1$. Additionally, we place of copy of this frame on the nearest *focus* within ρ. This simulates the phenomenon that gestures sometimes do not synchronize with their lexical affiliate like in "destroy an [entire planet]" where a Wipe gesture is performed on the bracketed part, although "destroy" is the lexical affiliate. The added gesture frame is represented by an arc that stretches across s, indicating the gesture's temporal position.

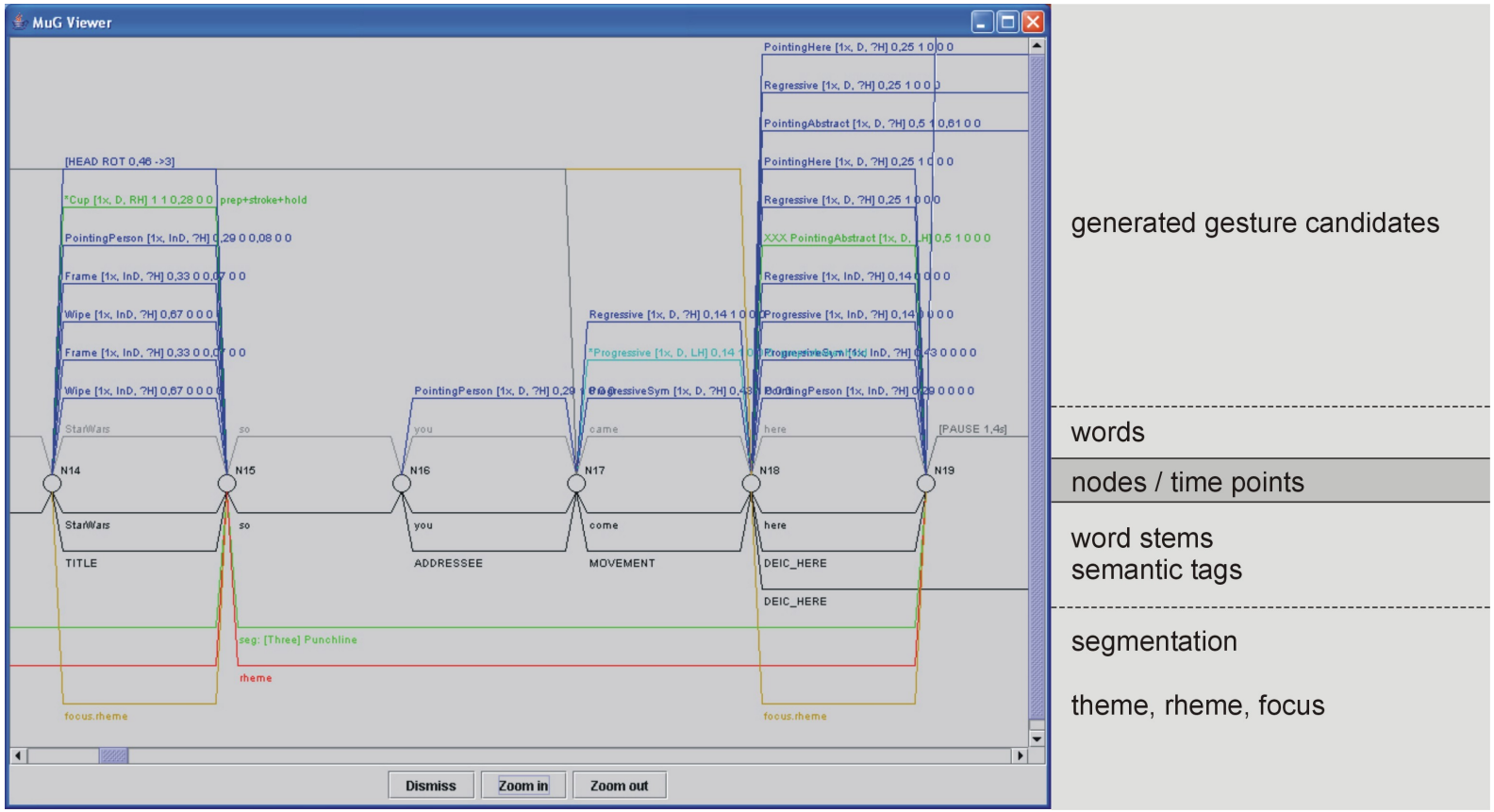

Figure 5.2: Multimodal generation graph containing input text with theme/rheme/focus tags (lower arcs). Upper arcs represent generated gesture frames.

5.2.2 Gesture Selection

In the next step, a path of non-overlapping gestures is selected using the maximum likelihood approach [17]. We select the gesture sequence that maximizes the probabiliy that this sequence $(g_0, \ldots, g_N)$ is observed, where the g_i represent the lexemes of the gestures. Using the Markov assumption, we approximate $\prod P(g_i|g_0, \ldots, g_{i-1})$ by $\prod P(g_i|g_{i-1})$ and use an interpolation of the estimated probabilities $\hat{P}$ from the gesture profile (Section 4.2) to smooth the estimation $\overline{P}$ of $P(g_i|g_{i-1})$:

$$\overline{P}(g_i|g_{i-1}) = 0.6\,\hat{P}(g_i|s) + 0.4\,\hat{P}(g_i|g_{i-1})$$

The next step determines the handedness of the gestures using a linear combination of estimated probabilities

$$\overline{P}(h_i|h_{i-1}) = 0.5\hat{P}(h_i|g_i) + 0.2\hat{P}(h_i) + 0.3\hat{P}(h_i|h_{i-1})$$

where the weights were empirically determined and the handedness is found by maximizing the probability of the handedness sequence $(h_i, \ldots, h_N)$.

Handshape is determined by consulting a lexicon where legal handshapes for each gesture are specified (e.g. pointing can be done with the index finger or the open hand). Handshape selection now follows the rule of economy: if the handshape of the previous gesture is a legal handshape for the current one, then keep it. Otherwise change handshape to a suitable one.

After this stage of generation we have an optimal gesture sequence of gesture frames where only lexeme, handedness and handshape are specified.

5.2.3 Creating Gesture Units

Combining multiple gestures to units is essential to achieve a fluent and continuous style of motion. We produce g-units by gradually merging gestures according to certain criteria. First, we take the first and the last gesture within a discourse segment and merge them with all in-between gestures to form a unit, i.e. adding a g-unit edge to the graph. Then, we cluster neighboring g-units by merging all units whose distance in seconds is below a threshold θ_{unit}. We found $\theta_{unit} = 1.5$ seconds to be a good value. The threshold could be made speaker-dependent: a high value produces large units with many gestures, a low value produces more isolated gestures.

5.2.4 Planning a Gesture Unit

For determining phase structure, positions and timing, we string together suitable samples from the GestureDB and let emerging constraints guide the determination of phases.

The phase structure (prep, stroke, hold etc.) of a gesture G_i depends on the temporal constraints of neighbouring gestures G_{i-1} and G_{i+1}. If there is enough space up front to perform a preparatory motion ($>$ .5 s) the gesture is assigned a preparation phase (which is always the case for the first gesture in a unit). If a gesture has a preparation, the positions of the gesture are unconstrained, so we select a random sample of the respective lexeme from GestureDB. If a gesture has no preparation, we find a sample whose start location best matches the end location of G_{i-1}. The chosen sample is used to specify positions, trajectory and type (s-gesture or h-gesture). To create multi-strokes for an s-gesture we consult the average number of multi-strokes, $\mu_{strokes}$, for gesture G_i. If $\mu_{strokes}$ exceeds a threshold we generate a random number of multi-strokes using the mean value and standard deviation. If there is not enough space between the gestures and $\mu_{strokes}$ exceeds a yet higher threshold, then gesture G_{i+1} is either moved back in time, where the speech-gesture offset's standard deviation is an upper bound on how far it can be moved, or eliminated in favor of G_i's multi-strokes. For all other cases where there is space between G_i and G_{i+1} we generate a hold between them.

Now the main stroke of G_i can be precisely timed with speech by aligning the end time of the stroke with the end time of the gesture's arc in the graph, which corresponds to the end time of the word(s) that triggered this gesture. From this time point we subtract the speaker's average offset ΔT_{end} for this lexeme. We hypothesized that it is more important that gesture and speech *end* synchronously than that they start synchronously. To compute the timing of after-strokes we align all after-stroke end times with word end times enforcing a minimum duration for each after-stroke. This proved to be an efficient way of achieving after-stroke synchronization. For hold gestures we use a similar method but align *start* times of word and hold. Looking at the resulting animations with a virtual character we found that although the timing was similar to the original speakers' timing the gestures always seemed a little too late. Human speakers can supposedly vary the timing of their gestures with great flexibility because movements of the whole body, and especially the face, all contribute to gesture-speech synchronization. Since our virtual character has a comparatively limited expressiveness we make our gesture timing more conservative by subtracting a general offset of 0.3 seconds for main strokes and 0.12 seconds for after-strokes.

Using this algorithm we generate the following types of gestures: stroke, prep-

stroke, prep-hold, prep-stroke-hold and stroke-hold, where all the strokes can be multiple strokes. As the g-unit's last gesture must by definition have a retract phase we have to determine to which rest position the hands return to. Observing the video material we determined three different rest positions: hands "at side", "in pockets", and "clasped". While the retraction after a unit could be modeled probabilistically we resorted to simple rules that work on the distance to the following g-unit: if small, retract to "clasped"; if medium distance retract to "at side"; if far retract to "in pockets".

5.3 Body Movement and Gesture Script

For creating body and head rotations we use simple rules taken from our observations of the subjects JL and MR, based on their discourse structure. JL seems to follow a clear pattern. His monologue routine consists of a series of jokes, each taking about 10 seconds. Before a joke he turns right (JL's viewpoint), probably to read the teleprompter, and before and after the punchline he often turns left to address the band-leader. Since MR is participating in a discussion, he rather turns to the person he addresses and stays there for a while. Both subjects anticipate their body rotations by turning their head a little earlier. JL also follows a gaze pattern in trying to distribute his gaze uniformly across the audience. We included these behaviours in our system in order to make the agent look more alive. The body rotation rules conform to empirical findings that changes in body postures occur most often at boundaries of discourse units ([3] and [41]).

The final graph is written to a linearized gesture script containing the following data: head rotations, body rotations and gesture units which contain one or more gestures and have a retract position specified. For each gesture the script specifies: lexeme, handedness, handshape, type (e.g. prep+stroke+hold or prep+hold), stroke/hold start time, multi-stroke start times, overall duration, number of strokes, gesture start location, gesture end location (only for s-type). An excerpt from a gesture script is given in Appendix A.

6 Animation

The role of the animation system is to take the gesture script as input and produce a final character animation sequence. It does this by augmenting the data provided by the gesture script, mapping this complete set of data to a form that can be animated, and then producing an either kinematic or dynamically simulated animation.

The animation system used is an extended version of the one described in Neff and Fiume [32], which is built on top of the DANCE framework [42]. Significant additions to the system include the use of offset layers and a set of augmentation processes that produce detailed animation specifications from the gesture script. The focus of this section will be on the new aspects of the system and how they are applied to the gesture animation task. The reader is referred to [32] and [30] for other details on the system.

The system also generates facial animation for lip sync [7] and additional facial movement such as eyebrow raises on stressed phonemes [1].

6.1 Underlying Representation

The prep-stroke-hold structure of gestures maps easily to animation keyframes. Our underlying movement representation, therefore, is analogous to a keyframe system. Every DOF in the character's body has its own track, partial body poses are stored at particular points in time and transition functions (cubic Hermite curves embedded in space and time) control interpolation between these poses.

Our representation extends a traditional keyframe system in two ways. First, we support *offset layers* and second, we support non-DOF tracks that can be used to adjust real time processes. Offset tracks are separate tracks that are added to the main track to determine the final value of a DOF. While offset layers are a

familiar tool for making low-frequency edits to motion capture data that preserve the high frequencies of the motion [48], we employ them differently. We use them to layer different motion specifications together and to add detail to our motion. In addition, some parts of the body are controlled by real-time processes for gaze-tracking and balance control. The desired constraints for these process are specified on separate tracks in the underlying representation using the same key and interpolation function primitives, allowing them to be continuously varied.

6.2 Data Augmentation

The gesture model needs sufficient control to correctly align gestures with speech and to reflect the key idiosyncrasies of a speaker's gesturing style. Gesture data is divided between the gesture model and animation engine in an effort to strike a balance between (A) the need for the gesture model to control the motion, (B) the desire to minimize the work required to annotate video for building the gesture model, and (C) the desire to allow the animation engine, which contains the relevant domain knowledge, to control the low-level aspects of motion production. The gesture script presents a minimal description of the required movements that captures these key definitional aspects. As summarized in Section 5.3, this includes the gesture lexeme, the end time and duration of the stroke, which hands are used for the gesture and discrete spatial locations of the hands. The animation system must augment this sparse representation, filling in more detailed data and adding important nuance. The process is one of refinement, continually adding more detail to improve the gesture rendering. Such an approach also allows for workload management as the animation can be generated after minimal augmentation, but adding more data to the animation lexicon will improve the quality of the animation.

The animation system performs a range of operations during data augmentation. It will:

- complete timing information
- deal with spatial conflicts due to the sparcity of spatial sampling
- add necessary definitional information for different gesture types
- add character specific variation.

These items will be detailed below. Character specific data includes global character properties [32], such as a default posture or tendency to start movements quickly, as well as variations on how a particular gesture is performed. Significant

use of two techniques is made to augment the initial motion framework: *keyframe infilling* and the addition of *micro-keys*. Micro-keys are parameters that define partial body poses and can be layered on top of existing keyframes. Keyframe infilling is a process by which new keyframes are generated at locations in between the existing key-frames. These can be micro-keys (partial specifications) as well.

6.2.1 Completion of Timing Data

The gesture script specifies end times and durations for strokes as well as hold durations and start times for body rotations. The rest of the required timing data is determined by the animation planner. The start time for prep movements is defined as:

$$prepStart = max(strokeStartTime - defaultPrepTime,\ lastStrokeEndTime) \tag{6.1}$$

where $defaultPrepTime$ is currently 0.4 s for preps within a gesture unit and 0.8 s for preps following a rest pose as these will have a longer distance to travel. A similar rule is used to determine the duration of the transition to rest poses where the time between the end of the last hold phase and the next stroke must accommodate both the transition to the rest pose and the subsequent prep phase. If there is enough time, 0.8 s is allowed for each. Otherwise, the time is split between the two movements. Head movements are given a duration uniformly distributed between 0.2 and 0.3 s or until the start time of the next movement if it is sooner. Body rotations are given a duration between 0.5 and 0.6 s.

6.2.2 Spatial Augmentation

To ease the annotation task, a relatively coarse spatial discretization is used to record gesture locations(Table 4.1). When generating animation, the system will place the hands at the middle of the spatial buckets corresponding to their discrete location tags from the annotation. While generally sufficient, this can occasionally lead to problems during animation where either the two hands may overlap, or the hand is not moved for a subsequent gesture when it should be. A small number of gestures, such as a wipe, feature the hands crossing over each other. When both hands cross, they may be annotated with the same discrete tags and a small two handed separation distance that will cause them to be placed in the same location when the data is used for animation. This is a rare occurence, but must be dealt with. A slightly more common occurrence is for subsequent gestures to be given

the same spatial location when there should actually be a small movement between them (even though both might still be in the same spatial bucket). This follows directly from the lack of resolution in the discretization. Both of these cases are automatically detected and offsets are applied to the hands to correct them. A vertical offset is used to separate overlapping hands and a downbeat is applied to sequential gestures with identical locations.

6.2.3 Additional Gesture Data

Handedness, spatial location and timing information are not sufficient to define most gestures. Additional information must be added to the gesture description from the animation lexicon, which contains data for each gesture the system can produce. Forearm rotation and the two wrist DOFs act to orient the hand. This is definitional for most gestures. For instance, a cup or a shrug will always have the palm facing up; a dismiss will have the palm facing down and end with a bent wrist. The system allows this data to be specified either in terms of joint angles or by giving orientation constraints defined in world-space or character chest space. Such data is an example of a micro-key – defining a limited number of DOFs and combining these with the previously defined key data.

Some gestures require a particular warp to the motion envelope. For example, a wipe gesture in which the hands are moved from the centre out to the side will generally feature an acceleration throughout the movement and will not look correct with an ease-in ease-out transition. Such changes to the transition can be achieved by specifying in the animation lexicon either a warping of the interpolation curve or a change in joint tension.

For some gestures, posture deformations are important. For instance, the chest will normally be opened (backward movement of the collar bones) during a wipe. The micro-key approach is used again, where only the specific parts of the posture that need to be varied are defined, and the rest will assume default values. We use the reduced parameter representation presented in [33] for defining these posture changes. Posture variations appear to be more related to the specific individuals than the other additions made to the gesture specification above, and are hence more likely to be customized.

The system automatically varies collar bone angles based on the gesture height. The form of after-strokes, which follow the main stroke, is also defined in the animation lexicon and added as part of the augmentation process. Both of these issues will be discussed below, in Sections 6.4 and 6.5.3 respectively.

Channel	*Description*
Body	Defines main body pose with reach targets.
Hands	Varies hand poses.
Rotation	Defines full body rotation.
Offset	Offsets are used to curve motions in space and vary elbow angles in after-strokes.
Gaze Direction	Moves a "look-at" target.

Table 6.1: Animation script channels and what they control.

6.3 Specification of Data

Animation data is specified through a set of independent channels that are then layered together to create the final motion. This allows different components of motion that all contribute to body pose, such as on-going body rotations and gesture specific posture changes, to be modeled separately, but then combined in the final animation. The channels are recorded in the *animation script*, which is an intermediate representation used to map the gesture script data to the underlying keyframe representation. The data contained in the channels is summarized in Table 6.1. The main framework of the motion is specified on the "Body" channel, which defines wrist constraints and gesture specific body poses. Characters will often turn to look at different people or locations while talking. This is modeled on the "Rotation" channel. "Offsets" are used to add curvature to motion, to add posture deformation and also during after-stroke specification. Hand shape is controlled on the "Hands" channel. "Gaze Direction" is used to model the character's continuous head movement. Global character data, such as a default posture, is blended with the properties contained on the other channels.

Body rotation is discretized into three directions (left, right, front) and gaze direction is discretized into five locations (left, left-front, front, right-front, right). The exact values of these locations are specified for each character based on the video corpus, with JL having larger rotations. Body rotations are accomplished by a combination of a pelvic rotation with opposing knee bend and a rotation of the abdomen and chest spine joints. These rotations are offset in time by ten % of the rotation's duration to give a more natural flow to the motion. They are specified on offset layers and blend with other posture deformations.

The IK constraints on the hands are body relative both in the annotation process and in the animation process. This allows the hands to move naturally with the body as it rotates and bends, supporting the superposition of different movement aspects related to posture change and gesture location.

6.4 Pose Calculation

The system uses a combination of keyframe and continuous motion generation approaches: discrete pose calculation with interpolation is used for gesture generation, and continuous IK for balance control and gaze tracking. Rather than a monolithic IK system that solves for an entire posture, we use a set of simple, analytic IK routines that are each responsible for a portion of the body. Lower body movement, including knee bends, pelvic rotation and balance control, is based on an analytic lower body IK routine and feedback based balance adjuster. To avoid stability problems in dynamic simulation, springs are used to hold the character's feet in position and prevent him from falling. Analytic routines are used for arm positioning, and aesthetic trunk constraints are blended together. The pose solver is similar to that described in [33], except that we disable the optimization routine described there as we are not using spatial constraints to deform the character's posture in this work. We also augment that system with automatic collar bone adjustments and add local IK routines to achieve hand and wrist orientation constraints.

The discrete poses used to generate the gestures are calculated in a preprocessing phase. The DOF values for each pose are determined in four steps. The first step calculates the posture of the spine and collar bones. To do this, the posture constraints for the character's default posture and any posture adjustments specified for the gesture in the animation lexicon are blended together. In addition, we introduce an automatic collar bone adjustment algorithm that offsets the shoulders up or down based on the location of the reach constraint. The base offset for each of the constraint heights is shown in Table 6.2. These base offsets are multiplied by a character specific gain value, which is 1 by default. Collar bone adjustment is important for increasing the naturalness of the arm movement. In step 2, the arms are positioned to meet the wrist constraints specified in the gesture description. The constraint locations are body relative as described below. The arm swivel angle is rotated to meet the inclination constraint in step 3. Finally, if world or chest space orientation constraints have been placed on the forearm and hands, these are solved for using standard trigonometry.

The gesture targets are defined to be relative to the current orientation of the body (cf. Table 4.1), so a shoulder height target will remain at shoulder height as the character hunches over. Although body relative, the constraint values that are actually used are computed in world space. Each height target, such as "shoulder", "chest" or "abdomen", has a defined height within a particular limb in the skeleton. The world location of that point is computed for the current skeleton deformation, defining the world height value. The radial distance from the char-

Constraint Height	*Base Offset (deg)*
above-head	-5
head	-3
shoulder	-1
chest	2
abdomen	3
belt	5
below-belt	7

Table 6.2: Base offset angles applied to the collar bones. These are multiplied by a character specific scale factor.

acter's centre and the distance in front of the character's body both lie on the world space horizontal plane with the given height value. The distance from the character's body ("touch", "close" etc.) are defined relative to the body part the gesture is in front of. The radial inclination, however, is defined relative to the chest. This means that a "front" radial inclination will be in front of the chest even if it is at belt height and the torso is rotated. These definitions perform well with the annotator's expectations of the mark up scheme.

Continuous motion generation involves two IK routines which operate during the animation process. The lower body chain from foot to foot is solved at each time step using the routine mentioned previously, combined with balance adjustment. Gaze tracking uses analytic routines to solve for the axial and tilt orientation required to have the character look at a given point. Axial rotation is distributed between the head and neck joints and tilt is applied solely to the head. A gain factor controls how far the head moves between staring straight ahead and staring at the look-at target. A gain of 50 % appears to provide natural head movement.

6.5 Keyframe Refinement

In addition to the layering of micro-keys on top of existing keys, as described in Section 6.2.3, some gesture attributes require the creation of additional keys, as detailed here.

6.5.1 Path-in-Space

Whether a movement follows a straight or curved path in space is an important expressive property. In our previous work, we did not model this property [32]. Chi *et al.* [6] in the EMOTE model represent it by varying the trajectory of the arm end-effector and also provide three different interpolation spaces: interpolating joint angles, end effector position or elbow position. Similarly, Kopp and Wachsmuth [25] use guiding strokes in space that allow the curvature of a motion to be controlled. Unlike the previous approaches, we achieve satisfying curved motions by working in joint space, using offset curves, rather than working in world space. This approach is simple and avoids the need to generate world space trajectories which may not be oriented with the most natural path for the motion.

There are two main types of curvature we need for our gesture lexicon: point-to-point curvature, where a single stroke follows a curved path, and continuous circular movements for gestures like progressives. The latter case will be discussed below. By default, movements between two points in our system will produce a basically straight path[1]. A curve can be added to the motion by introducing an offset perpendicular to the path of the movement that starts at zero, peaks near the middle of the movement, and returns to zero. For example, a "cup" movement that has the hand up and a largely horizontal trajectory, can be curved by adding an offset to the elbow bend as shown in Figure 6.1. We normally apply these offset keys to the elbow. Larger amplitude offsets will produce a higher curvature motion.

6.5.2 Progressives

A *progressive* is a cyclical movement in which the forearm and hand are moved in a circular loop in front of the chest, first coming towards the body and up, then out from the body and down. *Regressives* consist of the same motion rotated in the opposite direction. The specification of a progressive in the gesture script indicates which hand(s) are involved, the starting and ending location of the wrists and the number of repetitions. The cyclic movement definitional for the gesture must be fitted into this framework. This is done using keyframe infilling, extending the method for basic motion curvature above. To our knowledge, previous embodied conversational agent systems have not modeled progressives.

Considering a 2-DOF pendulum, if the x and y angles of the pendulum are both

[1] Within the limits of basic quaternion interpolation of the shoulder, which can introduce some warping to the path.

(a) Straight Trajectory (b) Curved Trajectory

Figure 6.1: Comparison of a straight and curved trajectory for the same *cup* gesture.

varied according to sine waves that are ninety degrees out of phase, then the end of the pendulum will move along a circular path. This is the basic idea used to generate progressives in the system. x and y rotations of the forearm are achieved by flexing the elbow (y) and rotating around the axis of the upper arm (x). Doing this out of phase creates a circular movement. The hand is also cycled by applying a similar process to the two wrist angles. We decompose the movement of the x and y DOFs of the forearm into two components, one that represents the cyclical movement of the progressive and one that provides the translational movement needed to meet the wrist position constraints. These are then superposed to generate the final motion.

The following process is used to create a progressive:

```
(1)  Determine the amplitude of the circular movement
(2)  Determine the time that each infilled keyframe must occur at
     For each infilled keyframe
(3)       Determine the x rotation and corresponding hand rotation
                          - OR -
(4)       Determine the y rotation and corresponding hand rotation

(5)       Add the keyframe to the system

(6)  Add an offset curve to account for elbow translation
```

Consider the circle in Figure 6.2 as representing the path of the wrist in space due to the rotational component of the progressive. The size of this circle will be determined by the amplitude of the x and y rotations. We define a *total amplitude* of the rotation, a, which measures the total angular distance of the motion and is calculated by finding the mean

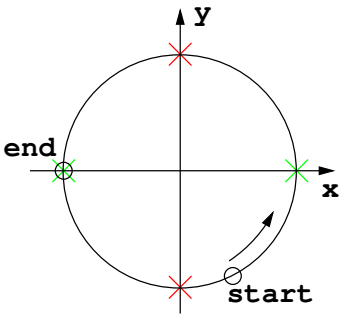

Figure 6.2: The path traced by the wrist during a progressive, ignoring translatory motion. Keyframes are located at the coloured crosses. The movement travels between the two small circles, along the arc of the large circle.

pose between the start and end positional constraints and determining the maximum of the total x and the total y rotation needed to reach the start and end positions from this pose. This is used to determine the progressive's amplitude α as follows (step 1):

$$\alpha = \max(MIN\_AMPLITUDE,\ m\frac{a}{2n}) \tag{6.2}$$

where n is the number of repetitions of the movement, m is a character specific multiplier which by default is 1 and MIN_AMPLITUDE is a minimum rotation that is provided to still create a progressive if the end constraints are the same. This relation gives the rotation an appropriate scale for the amount of spatial distance that will be covered during the movement.

The keyframes for the x DOF and corresponding hand movement will be placed at the green extrema in Figure 6.2 and the y keyframes at the red extrema. These are connected with simple ease-in/ease-out curves which provide an acceptable approximation to a sinusoid. By default, progressives start at the position marked "start", $5/6\pi$ for counter-clockwise rotation, and end at the location marked "stop", $-1/2\pi$ for counter-clockwise rotation. Each additional repetition adds a loop of the circle. The time of the keyframes is determined by tracing the movement around the circle in either the clockwise or counter-clockwise direction, as appropriate, and at each axis crossing (keyframe location) determining the amount of angular distance that has already been covered as a fraction of the total amount that must be covered (step 2).

Steps 3 and 4 determine the angles used at each keyframe (note that each keyframe has data for either x or y, but not both). Different angle representations will require different approaches here. We employ Euler angles at the elbow, so the values used for the y keyframes are simply $\pm\alpha$ as appropriate. The shoulders are more complicated as they are represented with quaternions and hence do not have a separate component corresponding to the x rotation. In determining these keys, we combine the rotational component of the progressive and the overall translatory aspect of the movement. We first update the

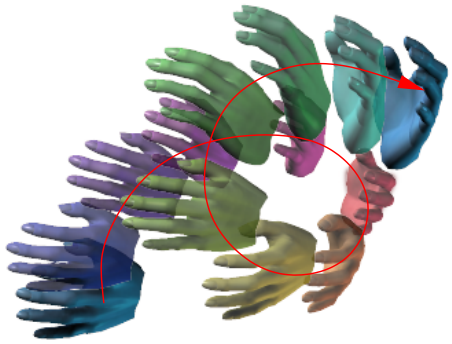

Figure 6.3: The path of a hand during a large progressive with a diagonal translatory component. The colour gradation is used to show the progress of time.

location of the upper arm:

$$q_i = slerp(q_0, q_1, p) \tag{6.3}$$

where q_0 is the quaternion satisfying the initial constraint, q_1 for the final constraint, q_i is the infill quaternion we are calculating and $p \in [0,1]$ is a progress variable indicating how far we are between the beginning and end of the progressive. This calculation corresponds to the translatory portion of the progressive. The rotational component is achieved by rotating q_i around a vector aligned with the axis of the upper arm by $\pm\alpha$ as appropriate.

In step 5, these keyframes are added to the underlying representation. The translatory component of the elbow movement has not been accounted for yet. This is done by adding an offset curve for this DOF that spans the translation (step 6).

An example of a fairly large progressive is shown in Figure 6.3. Notice that there is a diagonal translatory component to the motion of the gesture as well as the core circular motion of the progressive. A trace of the hand's path shows a circle stretched in time.

6.5.3 After-Strokes

A multi-stroke consists of the main stroke phase followed by a number of smaller repetitions known as *after-strokes*. There are at least two categories of after-strokes. The first consists of essentially continuous, rhythmic hand waving at the end of the stroke. MR frequently uses such gestures. The second has the same prep-stroke-hold structure as the main gesture. The hold period in these after-strokes is particularly important. Consider the after-stroke associated with a "dismiss" (a downward movement of the forearm and hands, palm facing down). It consists of a prep phase involving a slight upward movement

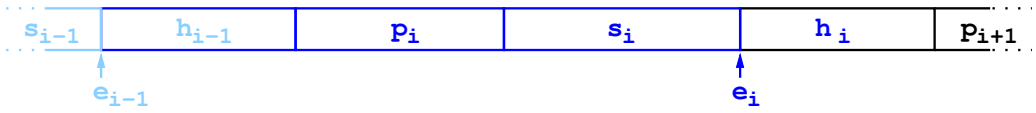

Figure 6.4: The end times e_{i-1} and e_i are constraints specified by the gesture generation module. The remaining timing of the after-strokes must be calculated to fit into these intervals. Here h indicates a hold, p a prep and s a stroke.

of the wrist and bend of the hand, followed by a downward stroke. When such a movement is repeated without a hold phase after the stroke, it disintegrates (disappears) into hand waving. The hand appears to bounce up from the end of the motion and the definitional downward aspect of the movement is lost without the necessary pause at the stroke end. The resulting after-stroke provides a completely different impression, conveying a different meaning.

Most data associated with after-strokes is local as they are normally rapid movements that are confined to the wrists and forearms. The animation lexicon accepts any subset of the following data to define an after-stroke pose: offset to vertical or horizontal wrist positions, forearm rotation, wrist rotation (2 possible DOFs) and an offset to the elbow bend. This data is defined separately for each of the two movements making up an after-stroke (nominally, prep and stroke). It is only defined for one arm and mirrored to the other. The system uses this data to create micro-keys that are then added to the copies of the main stroke keys as discussed above.

After-strokes are created by keyframe infilling. Copies of the stroke keyframe from the end of the main stroke are made and used as the basis for both the prep and stroke poses of each after-stroke. This is done as we wish to add small, local variation to the end position of the main stroke, rather than copying the often larger spatial movement of the main stroke. The timing of these keyframes is illustrated in Figure 6.4 and calculated as follows. The end time of the previous stroke (main or after-stroke) is e_{i-1} and the end time of the after-stroke being added is e_i. These are the key alignment markers with the spoken text and must be maintained. Note that in this window, we must fit a hold associated with the *preceeding* stroke, as well as prep and stroke phases associated with the current stroke. The hold phase will have zero duration in the case of continuous waving motions. The animation lexicon defines the percentage of time that should be spent on each of the prep, stroke and hold phases. These are used to determine the duration of p_i and s_i as a portion of $e_i - e_{i-1}$. In addition, h_i is calculated as a percentage of $e_i - e_{i-1}$. Finally, h_{i-1} is calculated using the same percentages, but based on the duration of $e_{i-1} - e_{i-2}$. If h_{i-1}, p_i and s_i can all fit within the duration $e_i - e_{i-1}$, p_i and s_i are used as calculated and h_{i-1} is extended to occupy all remaining time. If, however, there is not enough time to accommodate h_{i-1}, the duration of h_{i-1} is taken to be $(h_{i-1} + h_i)/2$ and p_i and s_i are refactored to fill the remaining time.

After-stroke durations may vary widely and we wish to reuse a single definition for each after-stroke of a given lexeme. To avoid unnatural movements when after-strokes are very short, limits are placed on the average spatial and angular velocity of after-strokes for kinematic animation (these limits are unnecessary in the dynamic case). If these limits will be exceeded, the spatial or angular range is reduced so that the average velocity is not exceeded.

6.6 Physical Simulation

Physical simulation can improve the realism of the resulting gestural animation in several ways. First, it will smooth the motion. Second, there is very little basis in the collected data for providing small torso deformations that people normally make while gesturing. Simulation allows the transfer of force from rapid arm movements into the torso which can cause small deformations and improve the realism of the motion. Third, due to the damping in the model, simulation serves a regulating function and will limit the speed of movements with unreasonably high velocities. Finally, simulation can add small end-effects to the motion, such as pendular arm sway when a character brings his arms to his side or passive movements of the fingers.

When computing physically simulated animation, we use a controller based approach whereby an actuator is placed at each DOF which calculates the torque required to move the character towards the desired configuration. We use an antagonistic formulation of proportional-derivative control, following [31]. The control law is written as

$$\tau = k_L(\theta_L - \theta) + k_H(\theta_H - \theta) - k_d\dot{\theta}, \tag{6.4}$$

where τ is the torque generated, θ is the current angle of the DOF and $\dot{\theta}$ is its current velocity. θ_L and θ_H are the low (L) and high (H) spring set points which serve as joint limits, k_L and k_H are the corresponding spring gains, and k_d is the gain on the damping term. The tension T or stiffness of the joint is taken as the sum of the two spring gains: $T = k_L + k_H$.

When a given transition for a specific DOF begins, gain values for the current angle and the desired angle are computed and the transition is affected by interpolating between these values. The torques generated by gravity are calculated using the current state of the character and an estimate of the end state, and the gain values are computed to compensate for these torques. This allows joint tension to be varied during a movement while still ensuring joint positioning that is accurate, at least at steady state.

In our system, offset tracks are summed with the main track to produce final control values. In kinematic simulation, each track contains angle data that can be directly added. In physical simulation, we add the gains. Given a function $C(\theta) = (k_H, k_L)$ which computes gravity compensated gains for a given DOF value θ, the rules summarized in Table 6.3 can be used to compute gains on each track.

	Main Track	*Offset Track*
Initial Gains	$(k_H, k_L) = C(\theta - \theta_{offset})$	$(k_H, k_L) = C(\theta) - C(\theta - \theta_{offset})$
End Gains	$(k_H, k_L) = C(\theta_{main})$	$(k_H, k_L) = C(\theta_{offset2} + \theta_{main}) - C(\theta_{main})$

Table 6.3: Equations for calculating the initial and end gains for each transition for the main and offset tracks.

These rules follow from Equation 6.4 and the gains are calculated at the start of the motion. θ is the current angle for the DOF at this time and θ_{offset} is the current desired offset angle. For the end point gains, θ_{main} is the desired value on the main track at the end of the transition and $\theta_{offset2}$ is the desired offset value at the end of the transition. These values are estimated based on the transition curves associated with the DOF and offset tracks. The tension is kept the same for each component during a transition, but the start and end times of the main and offset curves do not need to be the same.

Aside from the balance problem which we mitigate by using springs to hold the feet in position, one of the main difficulties encountered with controller based simulation is setting the gain values appropriately to generate the desired visual appearance. The inertia weighting technique presented by [49] provides a good initial estimate for joint gains. We augmented this by an automatic sampling procedure that takes repetitions of a prototypical movement and computes gain and damping values that would yield a specified overshoot for a given DOF (e.g. a two degree overshoot in elbow angle at the end of the transition). This yields tables of tension and damping values that are useful for fine tuning the parameters when required. This tuning process is done once per character and then used to generate all animations. During retraction phases, we relax the character's hands. The gains used for this are calculated based on the approach described in [35].

The rapid, time synchronized movements in gestural animation are a challenge to model using a proportional derivative control approach due to the damping in the system. The actuators include damping, which is important for producing realistic motion. However, as we wish to operate at relatively low-tension levels that will enable the movement to be enhanced by many of the benefits of physical simulation listed above, the damping in the system will introduce lag. This is particularly significant when you have short duration, high velocity movements with precise timing. The lag causes two problems: first, it means that the movements will be slightly slower and so will be behind their desired time constraints. Second, if the kinematic trajectories are used as the basis of the PD-control, the extent of the movement will be reduced in many cases as the movement will not have time to reach the desired end-point before the control trajectory changes direction. We must compensate for both of these effects.

To ensure that simulated movements satisfy the script timing, we moved the start time of all poses earlier in dynamic simulation. Empirical tests showed -0.12 s to be an appropriate offset. There are two potential ways to maintain the extent of the specified motion: the

trajectory curves could have their extent increased or the duration of the movements could be shortened and pauses inserted, allowing the actual movement to "catch-up" with the desired trajectory. For most cases, we use the latter approach. The pauses allow time for the motion to complete before a direction change begins. The update rule for the duration of strokes, d_s, is:

$$d_s = min(d_s * .8,\ max(.15,\ d_s - .3)) \tag{6.5}$$

and the update rule for the duration of preps, d_p, is:

$$d_p = min(d_p * .8,\ max(.1,\ d_s - .15))\ . \tag{6.6}$$

In the case of progressives, the continuous timing of the motion is particularly important, so we increase the extent of the transition curves rather than shortening the duration of the movement components. This is achieved by multiplying the angular span of the movement α by a factor of 1.6.

A variable time step Rosenbrock integrator [40] is used to compute the motion using simulation code from SD/Fast [16]. A 58 s MR sequence computed in 14 minutes and a JL generated sequence of the same length took 10.5 minutes on a 3 GHz Pentium 4.

7 Results

The accompanying video includes three pairs of example animations produced by our system for the two subjects. In the first example, we show for each subject a re-creation of a particular sequence of their video corpus. The gesture scripts for these sequences are created directly from the video annotation (Figure 1.2), and kinematic animation is used. These sequences validate both the fidelity of the annotation process and the ability of the animation system to generate the specified movement in the gesture script. A comparison for several gestures of a recreation of MR data to the original corpus is shown in Figure 7.1.

The second examples use the same audio tracks as the first, but generate new gestures based on each speaker's respective gesture profile. Since this input was also part of the training data we subtracted the statistical data for these particular sample from the profiles before generating the new gesture scripts. These sequences were dynamically simulated. Although the generated gestures are not necessarily the same as in the original (or the recreation) the animations distinctly reflect the gesticulation patterns of the modeled individuals. The resulting animations present effective gesture timing, synchronized with the original audio, and gesture forms that are consistent with the modelled subjects. This offers validation for the generation model.

The final pair of examples show gesture sequences generated by each of the subject models and dynamically animated for a new passage of synthesized English text that is not contained in either video corpus. This demonstrates that our system can operate on novel text and is language-independent, since MR's gesture profile was built on German training data. The video also illustrates the role movement plays in creating the overall impression of an utterance. Even though the timing of the speech is unlike that of either subject, the resulting animations are characteristic of each speaker.

An accompanying video has been included that shows a side by side comparison of kinematic and dynamic animations. Small differences in timing are noticeable, but the overall synchronization remains intact. The differences between the kinematic and dynamic versions of the animation are subtle: the force of arm movements cause some secondary movement in the torso, there is slight pendular movement as arms move to a character's

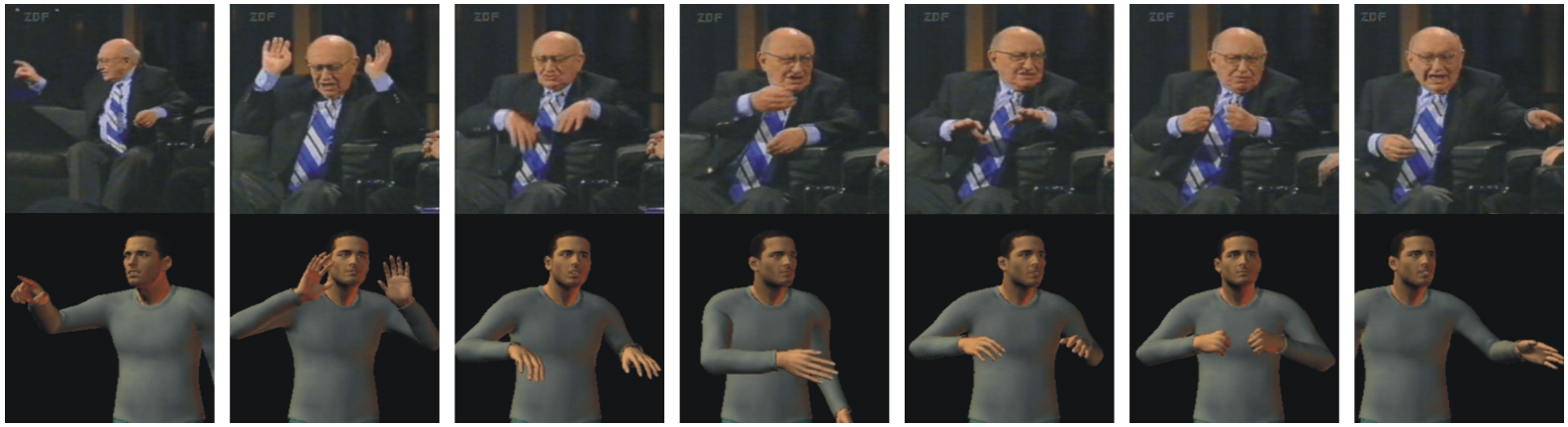

Figure 7.1: Frames of MR gesturing from the video corpus and frames of the same movement from animations recreated from the corpus. Note: In the fourth pair, MR is making a RH gesture and his left arm is at rest.

side, at times the trajectory is warped and there is small variation in the spatial path of the movements. Although subtle, we feel these effects help give the character a sense of "aliveness" that is less strong in the kinematic animation. This further validates the use of physical models and shows their relevance to synchronized gestural animation, where they have not previously been used. At the same time, there is much that can still be done to more fully take advantage of the power of physical models.

Finally, we include a side by side video comparison of the two models used on the "Star Wars" sequence. Frames from this are shown in Figure 7.2. This comparison nicely shows the style differences between the two models (JL vs. MR). Worth noting, not only does the system produce different gestures for each speaker, it also generates very different, yet still effective, timing patterns. For instance, the last pair of frames in the figure show a case where a gesture is generated for the JL model but not the MR model.

Overall, the animations show a high variation in gesture shape, good synchronization with speech and a nice overall flow of movement. High variation stems from using positional data from the GestureDB and from creating multiple strokes. The good synchronization validates our algorithms for aligning main stroke and after-strokes, using Steedman's concept of *focus* as an important gesture placement indicator. Finally, the overall flow is due to our introduction of the *gesture unit* as an organizational higher-level entity.

7.1 Validation

An evaluation study of our system was conducted that shows that the gestures produced by the system are recognizable as having style of the specific performer modeled[1]. For this study, 26 independent reviewers were recruited, aged 24 to 46; 6 female and 20 male, all non-expert in the field of gesture modeling and/or animation. In a learning phase they were shown video clips of the original performers, JL and MR. Two clips of each performer were used, about 5 minutes in total, and these clips were outside the training corpus used for our statistical models. The order of the clips was varied across subjects to avoid order effects. In the first test (Test 1) we showed them one video clip of generated gestures and asked them which original performer the gestures were modeled on. The JL model was used for half of the subjects and the MR model for the other half. The animations for the novel "Star Wars" text were used in the experiment (these are the last two clips discussed above) which are obviously outside our training corpus. Afterwards, in the second test (Test 2) we showed a side-by-side clip of both variants of the "Star Wars" clips (modeled on JL + modeled on MR) and asked subjects to decide which character was modeled after which performer (JL and MR). The result of Test 1 was that subjects selected the correct original performer 69 % of the time, which is significantly above chance ($t(25) = 2.083; p < .05$). The result of Test 2 was that subjects correctly

[1]This experiment was done using a slightly earlier version of the system. The quality of the animations has subsequently been improved.

Figure 7.2: A matter of style: Different gestures were generated for contemporaneous frames of the “Star Wars” animations for JL (top row) and MR (bottom row).

assigned the original performer to the side-by-side characters in 88 % of the cases which is also significantly above chance ($t(25) = 6.019; p < .001$). In Test 1, there was no significant difference in recognizing JL compared to recognizing MR ($t(24) = .5708; p = .57$). The evaluation clearly shows that the produced animation reflects the style of a specific performer. This worked equally well for both performers. When putting the animations side by side, the discrimination task is even easier, as reflected by the higher scores. It should be noted that the selected clips were generated on synthetic text for which we did not model the timing and speaking pattern of either speaker. This makes the recognition task harder as these important aspects of personal style were absent in the stimuli, providing less information than would be present in a clip of either speaker.

8 Conclusion and Discussion

This work presents a system for generating believable gesture animations for novel text that reflect the gesturing style of particular individuals. It moves beyond previous approaches by creating a statistical model of particular individuals; modeling gestures at a high level of detail; modeling complex gestures, including highly variable after-strokes and progressives; building effective gesture units that flow well and synchronize effectively with speech; and using physical simulation to enhance the final animation. Gesture units are a particularly effective construct. We generate stretches of gestures and infer timing parameters from the interdependence of the gestures contained in one unit. This makes gesture flow much more natural as the gestures are connected by holds or directly succeed each other while positional parameters are fitted depending on the preceding gesture. We found the alignment of end times between gesture strokes and speech correlates performed well. Finally, we exploit Steedman's concept of "focus" to synchronize gestures not with the directly related part of the utterance (lexical affiliate) but with the focus. The successful timing this produces breaks the myth common in the literature of a gesture having to occur slightly before its semantic correlate.

It is worth highlighting the effective division of data between the gesture generation process and the animation process. The gesture generation system contains the data necessary to both synchronize gestures with speech and to capture the spatial gesturing pattern of particular individuals. The animation system supports the reuse of labour through the animation lexicon, hides many details of animation production from the gesture generation process and effectively augments the generation data to produce convincing gestural animation.

Automatically creating animations of talking characters that reflect a specific subject's style and will satisfy a human observer is a very challenging task, and much work remains to be done. First, although the annotation scheme is simple, elegant and effective, the process is currently labour intensive. We see significant opportunity for automation in the process. Research has already been published on tracking JL's face and hands [45], which can likely be extended for use in our workflow. Second, certain movements were avoided. For instance, handrub motions require a high quality collision model; iconic gestures need a deeper model of semantics. As well, the system should be extended to model

non-hand based gestures such as head points and shoulder shrugs without accompanying hand movement. Better models for torso engagement while gesturing are also worthwhile. Finally, we conjecture that better use of physical models can further improve the quality of the animations. People continuously modulate the tension in their bodies while moving, in a much more complex way than modeled here. Despite this, it is worth noting that this work demonstrates the usefulness of physics based animation to gesturing characters. Using physical simulation for motion generation offers the potential to unify character animation and secondary effects like cloth modeling within the physical simulation realm. This will help ensure that character motions exert reasonable forces on these secondary models and offers the potential to create a continuous representation from limb movement, to muscle and skin deformation, to cloth and hair.

Appendix A
Gesture Script Example

```
2h_distance.min=0.3130435160733115

BEGIN_BODY_ROTATIONS
  ROTATE_BODY right 10.073230361
  ...
END_BODY_ROTATIONS

BEGIN_HEAD_ROTATIONS
  ROTATE_HEAD right 9.673230361
  ROTATE_HEAD front 17.305170822
  ...
END_HEAD_ROTATIONS

#-------------------------------------------------------
BEGIN_G_UNIT
  BEGIN_GESTURE
    # start time = 8,704
    # Triggered by: "civil war" AGGRESSION 10,473 - 11,188
    lexeme=Fist    # from sample 32  (random)
    handedness=2H
    handshape=fist
    type=prep+stroke+hold
    # total stroke duration = 1,685
    stroke.trajectory=straight    # from sample 32
    hold.time.duration=0.4880431763966211
    stroke.time.duration=0.6936347179353073    # ran. offset 0,289
    stroke.time.end=9.897439842000004    # offset -1,291
    stroke.number=3
    mstroke.0.time.end=10.485929679
```

```
    mstroke.1.time.end=10.88846035
    stroke.position.start.2h_distance=0.6537408296950502
    stroke.position.start.height=shoulder
    stroke.position.start.distance=close
    stroke.position.start.radial=front
    stroke.position.start.inclination=normal
    stroke.position.end.2h_distance=0.8116517219742472
    stroke.position.end.height=chest
    stroke.position.end.distance=close
    stroke.position.end.radial=front
    stroke.position.end.inclination=normal
    # end time = 11,377
  END_GESTURE

  BEGIN_GESTURE
    # start time = 11,377
    # Triggered by: "galactic empire" TITLE 16,944 - 18,005 [init]
    # sync'd with Init words "rebel space"
    lexeme=Umbrella    # from sample 77  (random)
    handedness=LH
    handshape=open-rlx
    type=prep+stroke
    stroke.trajectory=straight    # from sample 77
    stroke.time.duration=0.2074158836033774    # ran. offset -0,013
    stroke.time.end=12.08391941    # offset -0,535
    stroke.number=1
    stroke.position.start.height=belly
    stroke.position.start.distance=normal
    stroke.position.start.radial=out
    stroke.position.start.inclination=normal
    stroke.position.end.height=belly
    stroke.position.end.distance=normal
    stroke.position.end.radial=out
    stroke.position.end.inclination=normal
    # end time = 12,084
  END_GESTURE

  ... # more gestures

  RETRACT_GESTURE pose=at-side
END_G_UNIT

... # more gesture units
```

Bibliography

[1] I. Albrecht, J. Haber, and H.-P. Seidel. Automatic generation of non-verbal facial expressions from speech. In *Proc. Computer Graphics International 2002*, pages 283–293, 2002.

[2] P. Boersma and D. Weenink. Praat: Doing phonetics by computer (version 4.3.14) [computer program]. Retrieved from http://www.praat.org/, 2005.

[3] J. Cassell, Y. Nakano, T. Bickmore, C. Sidner, and C. Rich. Non-verbal cues for discourse structure. In *Proc. Annual Meeting of the Association for Computational Linguistics 2001*, pages 106–115, 2001.

[4] J. Cassell, C. Pelachaud, N. Badler, M. Steedman, B. Achorn, T. Becket, B. Douville, S. Prevost, and M. Stone. Animated conversation: Rule-based generation of facial expression, gesture and spoken intonation for multiple conversational agents. In *Proc. SIGGRAPH 1994*, pages 413–420, 1994.

[5] J. Cassell, H. Vilhjálmsson, and T. Bickmore. BEAT: The Behavior Expression Animation Toolkit. In *Proc. SIGGRAPH 2001*, pages 477–486, 2001.

[6] D. M. Chi, M. Costa, L. Zhao, and N. I. Badler. The EMOTE model for effort and shape. In *Proc. SIGGRAPH 2000*, pages 173–182, 2000.

[7] M. Cohen and D. Massaro. Modeling coarticulation in synthetic visual speech. In N. Magnenat-Thalmann and D. Thalmann, editors, *Models and Techniques in Computer Animation*, pages 139–156. Springer, Tokyo, 1993.

[8] J. de Ruiter. The production of gesture and speech. In D. McNeill, editor, *Language and Gesture: Window into Thought and Action*, pages 284–311. Cambridge University Press, Cambridge, England; New York, NY, 2000.

[9] P. Faloutsos, M. van de Panne, and D. Terzopoulos. The virtual stuntman: Dynamic characters with a repertoire of autonomous motor skills. *Computers & Graphics*, 25(6):933–953, 2001.

[10] W. Finkler and G. Neumann. MORPHIX. a fast realization of a classification-based approach to morphology. In *Proc. 4. Österreichische Artificial-Intelligence-Tagung. Wiener Workshop - Wissensbasierte Sprachverarbeitung*, pages 11–19, 1988.

[11] S. Frey. *Die Macht des Bildes: der Einfluß der nonverbalen Kommunikation auf Kultur und Politik*. Verlag Hans Huber, Bern, 1999.

[12] B. Hartmann, M. Mancini, S. Buisine, and C. Pelachaud. Design and evaluation of expressive gesture synthesis for embodied conversational agents. In *Proc. 4th International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1095–1096, 2005.

[13] B. Hartmann, M. Mancini, and C. Pelachaud. Formational parameters and adaptive prototype installation for MPEG-4 compliant gesture synthesis. In *Proc. Computer Animation 2002*, pages 111–119, 2002.

[14] B. Hartmann, M. Mancini, and C. Pelachaud. Implementing expressive gesture synthesis for embodied conversational agents. In *Proc. Gesture Workshop 2005*, volume 3881 of *LNAI*, pages 45–55, Berlin; Heidelberg, 2006. Springer.

[15] J. K. Hodgins, W. L. Wooten, D. C. Brogan, and J. F. O'Brien. Animating human athletics. In *Proc. SIGGRAPH 1995*, pages 71–78, 1995.

[16] M. G. Hollars, D. E. Rosenthal, and M. A. Sherman. *SD/FAST User's Manual*. Symbolic Dynamics Inc., 1994.

[17] D. Jurafsky and J. H. Martin. *Speech and Language Processing*. Prentice Hall, 2003.

[18] A. Kendon. *Gesture – Visible Action as Utterance*. Cambridge University Press, Cambridge, 2004.

[19] M. Kipp. Anvil – a generic annotation tool for multimodal dialogue. In *Proc. 7th European Conference on Speech Communication and Technology (Eurospeech)*, pages 1367–1370, 2001.

[20] M. Kipp. *Gesture Generation by Imitation: From Human Behavior to Computer Character Animation*. Dissertation.com, Boca Raton, Florida, 2004.

[21] M. Kipp, M. Neff, and I. Albrecht. An annotation scheme for conversational gestures: How to economically capture timing and form. In *Proc. Workshop on "Multimodal Corpora" at LREC 2006*, pages 24–27, 2006.

[22] S. Kita, I. van Gijn, and H. van der Hulst. Movement phases in signs and co-speech gestures, and their transcription by human coders. In I. Wachsmuth and M. Fröhlich, editors, *Gesture and Sign Language in Human-Computer Interaction*, pages 23–35, Berlin, 1998. Springer.

[23] S. Kopp, T. Sowa, and I. Wachsmuth. Imitation games with an artificial agent: From mimicking to understanding shape-related iconic gestures. In *Proc. Gesture Workshop 2003*, volume 2915 of *LNAI*, pages 436–447, Berlin and Heidelberg, 2004. Springer.

[24] S. Kopp, P. Tepper, and J. Cassell. Towards integrated microplanning of language and iconic gesture for multimodal output. In *Proc. International Conference on Multimodal Interfaces 2004*, pages 97–104, 2004.

[25] S. Kopp and I. Wachsmuth. Synthesizing multimodal utterances for conversational agents. *Computer Animation and Virtual Worlds*, 15:39–52, 2004.

[26] W. C. Mann and S. A. Thompson. Rhetorical Structure Theory: Toward a functional theory of text organization. *Text*, 8(3):243–281, 1988.

[27] C. Martell. FORM: An extensible, kinematically-based gesture annotation scheme. In *Proc. 7th International Conference on Spoken Language Processing (ICSLP 2002)*, pages 353–356, 2002.

[28] D. McNeill. *Hand and Mind: What Gestures Reveal about Thought*. The University of Chicago Press, Chicago, 1992.

[29] G. A. Miller, R. Beckwith, C. Felbaum, D. Gross, and K. Miller. Introduction to WordNet: An on-line lexical database. *International Journal of Lexicography*, 3(4):235–244, 1990.

[30] M. Neff. Aesthetic exploration and refinement: A computational framework for expressive character animation, 2005. Ph.D. Dissertation. Department of Computer Science, University of Toronto.

[31] M. Neff and E. Fiume. Modeling tension and relaxation for computer animation. In *Proc. ACM SIGGRAPH Symposium on Computer Animation 2002*, pages 81–88, 2002.

[32] M. Neff and E. Fiume. AER: Aesthetic Exploration and Refinement for expressive character animation. In *Proc. ACM SIGGRAPH / Eurographics Symposium on Computer Animation 2005*, pages 161–170, 2005.

[33] M. Neff and E. Fiume. Methods for exploring expressive stance. *Graphical Models*, 68(2):133–157, 2006.

[34] M. Neff, M. Kipp, I. Albrecht, and H.-P. Seidel. Gesture Modeling and Animation by Imitation. Technical Report MPI-I-2006-1-005, Max-Planck-Institut Informatik, Saabrücken, Germany, 2006.

[35] M. Neff and H.-P. Seidel. Modeling relaxed hand shape for character animation. In *Articulated Motion and Deformable Objects (AMDO 2006)*, volume 4069 of *LNCS*, Berlin, 2006. Springer.

[36] T. Noma, L. Zhao, and N. Badler. Design of a virtual human presenter. *IEEE Computer Graphics and Applications*, 20(4):79–85, 2000.

[37] H. Noot and Z. Ruttkay. Gesture in style. In *Proc. Gesture Workshop 2003*, volume 2915 of *LNAI*, pages 324–337, Berlin; Heidelberg, 2004. Springer.

[38] N. S. Pollard and V. B. Zordan. Physically based grasping control from example. In *Proc. ACM SIGGRAPH / Eurographics Symposium on Computer Animation 2005*, pages 311–318, 2005.

[39] Z. Popovic and A. Witkin. Physically based motion transformation. In *Proc. SIGGRAPH 1999*, pages 11–20, 1999.

[40] W. H. Press, S. A. Tukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 2nd edition, 1992.

[41] A. E. Scheflen. The Significance of Posture in Communication Systems. *Psychiatry*, 26:316–331, 1964.

[42] A. Shapiro, P. Faloutsos, and V. Ng-Thow-Hing. Dynamic animation and control environment. *Graphics Interface '05*, pages 61–70, 2005.

[43] M. Steedman. Information structure and the syntax-phonology interface. *Linguistic Inquiry*, 34:649–689, 2000.

[44] M. Stone, D. DeCarlo, I. Oh, C. Rodriguez, A. Stere, A. Lees, and C. Bregler. Speaking with hands: Creating animated conversational characters from recordings of human performance. In *Proc. SIGGRAPH 2004*, pages 506–513, 2004.

[45] R. Tan and J. Davis. Differential video coding of face and gesture events in presentation videos. *Computer Vision and Image Understanding*, 96(2):200–215, 2004.

[46] R. Webb. *Linguistic Properties of Metaphoric Gestures*. UMI, New York, 1997.

[47] A. Witkin and M. Kass. Spacetime constraints. In *Proc. SIGGRAPH 1988*, pages 159–168, 1988.

[48] A. Witkin and Z. Popovic. Motion warping. *Proc. SIGGRAPH 1995*, pages 105–108, 1995.

[49] V. B. Zordan and J. K. Hodgins. Motion capture-driven simulations that hit and react. In *Proc. ACM SIGGRAPH Symposium on Computer Animation 2002*, pages 89–96, 2002.

Below you find a list of the most recent technical reports of the Max-Planck-Institut für Informatik. They are available by anonymous ftp from `ftp.mpi-sb.mpg.de` under the directory `pub/papers/reports`. Most of the reports are also accessible via WWW using the URL `http://www.mpi-sb.mpg.de`. If you have any questions concerning ftp or WWW access, please contact `reports@mpi-sb.mpg.de`. Paper copies (which are not necessarily free of charge) can be ordered either by regular mail or by e-mail at the address below.

Max-Planck-Institut für Informatik
Library
attn. Anja Becker
Stuhlsatzenhausweg 85
66123 Saarbrücken
GERMANY
e-mail: `library@mpi-sb.mpg.de`

MPI-I-2006-RG1-001	S. Hirth, C. Karl, C. Weidenbach	Automatic Infrastructure for ..... Analysis
MPI-I-2006-5-005	R. Angelova, S. Siersdorfer	A Neighborhood-Based Approach for Clustering of Linked Document Collections
MPI-I-2006-5-004	F. Suchanek, G. Ifrim, G. Weikum	Combining Linguistic and Statistical Analysis to Extract Relations from Web Documents
MPI-I-2006-5-003	V. Scholz, M. Magnor	Garment Texture Editing in Monocular Video Sequences based on Color-Coded Printing Patterns
MPI-I-2006-5-002	H. Bast, D. Majumdar, R. Schenkel, M. Theobald, G. Weikum	IO-Top-k: Index-access Optimized Top-k Query Processing
MPI-I-2006-5-001	M. Bender, S. Michel, G. Weikum, P. Triantafilou	Overlap-Aware Global df Estimation in Distributed Information Retrieval Systems
MPI-I-2006-4-007	O. Schall, A. Belyaev, H. Seidel	Feature-preserving Non-local Denoising of Static and Time-varying Range Data
MPI-I-2006-4-006	C. Theobalt, N. Ahmed, H. Lensch, M. Magnor, H. Seidel	Enhanced Dynamic Reflectometry for Relightable Free-Viewpoint Video
MPI-I-2006-4-005	S. Yoshizawa	?
MPI-I-2006-4-004	V. Havran, R. Herzog, H. Seidel	On Fast Construction of Spatial Hierarchies for Ray Tracing
MPI-I-2006-4-003	E. de Aguiar, R. Zayer, C. Theobalt, M. Magnor, H. Seidel	A Framework for Natural Animation of Digitized Models
MPI-I-2006-4-002	G. Ziegler, A. Tevs, C. Theobalt, H. Seidel	GPU Point List Generation through Histogram Pyramids
MPI-I-2006-4-001	R. Mantiuk	?
MPI-I-2006-2-001	T. Wies, V. Kuncak, K. Zee, A. Podelski, M. Rinard	On Verifying Complex Properties using Symbolic Shape Analysis
MPI-I-2006-1-007	I. Weber	?
MPI-I-2006-1-006	M. Kerber	Division-Free Computation of Subresultants Using Bezout Matrices
MPI-I-2006-1-005	I. Albrecht	?
MPI-I-2006-1-004	E. de Aguiar	?

MPI-I-2006-1-001	M. Dimitrios	?
MPI-I-2005-5-002	S. Siersdorfer, G. Weikum	Automated Retraining Methods for Document Classification and their Parameter Tuning
MPI-I-2005-4-006	C. Fuchs, M. Goesele, T. Chen, H. Seidel	An Emperical Model for Heterogeneous Translucent Objects
MPI-I-2005-4-005	G. Krawczyk, M. Goesele, H. Seidel	Photometric Calibration of High Dynamic Range Cameras
MPI-I-2005-4-004	C. Theobalt, N. Ahmed, E. De Aguiar, G. Ziegler, H. Lensch, M.A.,. Magnor, H. Seidel	Joint Motion and Reflectance Capture for Creating Relightable 3D Videos
MPI-I-2005-4-003	T. Langer, A.G. Belyaev, H. Seidel	Analysis and Design of Discrete Normals and Curvatures
MPI-I-2005-4-002	O. Schall, A. Belyaev, H. Seidel	Sparse Meshing of Uncertain and Noisy Surface Scattered Data
MPI-I-2005-4-001	M. Fuchs, V. Blanz, H. Lensch, H. Seidel	Reflectance from Images: A Model-Based Approach for Human Faces
MPI-I-2005-2-004	Y. Kazakov	A Framework of Refutational Theorem Proving for Saturation-Based Decision Procedures
MPI-I-2005-2-003	H.d. Nivelle	Using Resolution as a Decision Procedure
MPI-I-2005-2-002	P. Maier, W. Charatonik, L. Georgieva	Bounded Model Checking of Pointer Programs
MPI-I-2005-2-001	J. Hoffmann, C. Gomes, B. Selman	Bottleneck Behavior in CNF Formulas
MPI-I-2005-1-008	C. Gotsman, K. Kaligosi, K. Mehlhorn, D. Michail, E. Pyrga	Cycle Bases of Graphs and Sampled Manifolds
MPI-I-2005-1-008	D. Michail	?
MPI-I-2005-1-007	I. Katriel, M. Kutz	A Faster Algorithm for Computing a Longest Common Increasing Subsequence
MPI-I-2005-1-003	S. Baswana, K. Telikepalli	Improved Algorithms for All-Pairs Approximate Shortest Paths in Weighted Graphs
MPI-I-2005-1-002	I. Katriel, M. Kutz, M. Skutella	Reachability Substitutes for Planar Digraphs
MPI-I-2005-1-001	D. Michail	Rank-Maximal through Maximum Weight Matchings
MPI-I-2004-NWG3-001	M. Magnor	Axisymmetric Reconstruction and 3D Visualization of Bipolar Planetary Nebulae
MPI-I-2004-NWG1-001	B. Blanchet	Automatic Proof of Strong Secrecy for Security Protocols
MPI-I-2004-5-001	S. Siersdorfer, S. Sizov, G. Weikum	Goal-oriented Methods and Meta Methods for Document Classification and their Parameter Tuning
MPI-I-2004-4-006	K. Dmitriev, V. Havran, H. Seidel	Faster Ray Tracing with SIMD Shaft Culling
MPI-I-2004-4-005	I.P. Ivrissimtzis, W.-. Jeong, S. Lee, Y.a. Lee, H.-. Seidel	Neural Meshes: Surface Reconstruction with a Learning Algorithm
MPI-I-2004-4-004	R. Zayer, C. Rssl, H. Seidel	r-Adaptive Parameterization of Surfaces
MPI-I-2004-4-003	Y. Ohtake, A. Belyaev, H. Seidel	3D Scattered Data Interpolation and Approximation with Multilevel Compactly Supported RBFs
MPI-I-2004-4-002	Y. Ohtake, A. Belyaev, H. Seidel	Quadric-Based Mesh Reconstruction from Scattered Data
MPI-I-2004-4-001	J. Haber, C. Schmitt, M. Koster, H. Seidel	Modeling Hair using a Wisp Hair Model

MPI-I-2004-2-007	S. Wagner	Summaries for While Programs with Recursion
MPI-I-2004-2-002	P. Maier	Intuitionistic LTL and a New Characterization of Safety and Liveness
MPI-I-2004-2-001	H. de Nivelle, Y. Kazakov	Resolution Decision Procedures for the Guarded Fragment with Transitive Guards
MPI-I-2004-1-006	L.S. Chandran, N. Sivadasan	On the Hadwiger's Conjecture for Graph Products
MPI-I-2004-1-005	S. Schmitt, L. Fousse	A comparison of polynomial evaluation schemes
MPI-I-2004-1-004	N. Sivadasan, P. Sanders, M. Skutella	Online Scheduling with Bounded Migration
MPI-I-2004-1-003	I. Katriel	On Algorithms for Online Topological Ordering and Sorting
MPI-I-2004-1-002	P. Sanders, S. Pettie	A Simpler Linear Time 2/3 - epsilon Approximation for Maximum Weight Matching
MPI-I-2004-1-001	N. Beldiceanu, I. Katriel, S. Thiel	Filtering algorithms for the Same and UsedBy constraints
MPI-I-2003-NWG2-002	F. Eisenbrand	Fast integer programming in fixed dimension