

# ANIMATING AN AUTONOMOUS 3D TALKING AVATAR

Dominik Borer<sup>1</sup>, Dominic Lutz<sup>2</sup>, Robert W. Sumner<sup>2</sup> and Martin Guay<sup>2</sup>

<sup>1</sup>*ETH Zürich, Disney Research*

<sup>2</sup>*Disney Research*



## ABSTRACT

One of the main challenges with embodying a conversational agent is annotating how and when motions can be played and composed together in real-time, without any visual artifact. The inherent problem is to do so—for a large amount of motions—without introducing mistakes in the annotation. To our knowledge, there is no automatic method that can process animations and automatically label actions and compatibility between them. In practice, a state machine, where clips are the actions, is created manually by setting connections between the states with the timing parameters for these connections. Authoring this state machine for a large amount of motions leads to a visual overflow, and increases the amount of possible mistakes. In consequence, conversational agent embodiments are left with little variations and quickly become repetitive. In this paper, we address this problem with a compact taxonomy of chat behaviors, that we can utilize to simplify and partially automate the graph authoring process. We measured the time required to label actions of an embodiment using our simple interface, compared to the standard state machine interface in Unreal Engine, and found that our approach is 7 times faster. We believe that our labeling approach could be a path to automated labeling: once a sub-set of motions are labeled (using our interface), we could learn a prediction that could attribute a label to new clips—allowing to really scale up virtual agent embodiments.

## KEYWORDS

Conversational Embodiment, Interactive Conversational Agent, Parametric Body Motion

## 1. INTRODUCTION

Intelligent avatars that can talk and interact with people offer a natural and friendly way to interface with autonomous systems—may they be cars, televisions, food dispensers, phones, maps, and so on. While this vision is not new, machine perception and natural Language processing has evolved rapidly in recent years,

causing a re-visitation of this vision by various companies, each releasing voice-based assistants such as Alexa, Siri, or AliGenie and some embodied systems with digital displays such as Gatebox, Jibo and Baidu.

One of the main challenges with embodying an agent is annotating how and when motions can be played and composed together in real-time, without any visual artifact. The inherent problem is to do so—for a large amount of motions—without introducing mistakes in the annotation. To our knowledge, there is no automatic method that can process animations and automatically label actions and compatibility between them. In practice, a state machine, where clips are the actions, is created manually by setting connections between the states with the timing parameters for these connections, together with other state parameters such as body part mask, clip start and end time, as well as whether the motion is looping or not. For example, consider an action that has the arm waving in the air, while a new action requests a beat gesture that starts with the arm pointing on the floor. The sudden jump to the other motion will cause a visual artifact. Hence in this case the state machine would have a transition between both gestures, and the timing would be set to the end of the first clip, forcing the embodiment to finish the first clip before playing the next (assuming they have a compatible pose at their respective extremities).

Authoring this state machine for a large amount of clips requires repetitive labour for many actions, and can grow exponentially for certain types of behaviors that require transitions between them, as shown in Figure 1. Hence the possibilities of introducing mistakes is very high. In consequence, conversational agent embodiments are often left with little variations, and quickly become repetitive and predictable. In this paper, we address this problem with a compact taxonomy of chat behaviors, that we can utilize to simplify and partially automate the graph authoring process. We observed from casual dyadic conversations that people are mainly in stances, fidgets, gestures and transitions (between stances). Stances are synonymous to idle (e.g. arm on waist, or body weight on one side), fidgets are ticks and small subtle gestures, while gestures are more functional. The final element are the transitions between the stances such as changing the body weight to another side, or having a hand going from the waist to a shoulder.

With our taxonomy, we propose a more abstract and partially automated interface for the state machine authoring, where animation clips are dragged and dropped onto one of the action types (gesture, fidget, stance, transition), as shown in Figure 2. We asked an animator to create animations guided by our taxonomy, and to utilize our interface to populate the graph. In total, 152 animations were created, and labeling all the motions (or authoring the graph) was 7 times faster than using the traditional interface in Unreal Engine (see our comparison discussion for more details Section 7.1). We believe that our labeling approach could be a way to automated labeling via a bootstrapping phase: once a sub-set of motions are labeled (using our interface), we could learn a prediction that could attribute a label to new clips—allowing to really scale up virtual agent embodiments.

We additionally report on our best practices for embodiment decision making (Section 6, which we learned during a collaborative project for embodying an agent. We found that in practice agents operate at different levels of abstraction. For example, one agent might output high-level actions such as “say line 279”, or lower level commands such as “nod here”. To cope with different levels of abstraction, we designed a motion planner that can filter, as well as synthesize actions, if needed. Our planner takes sequences of abstract actions and outputs a sequence of specific actions. In the case of too few actions, the planner can generate natural sequences using probability distributions conditioned on the previous action, which we compute from video recorded human performances.

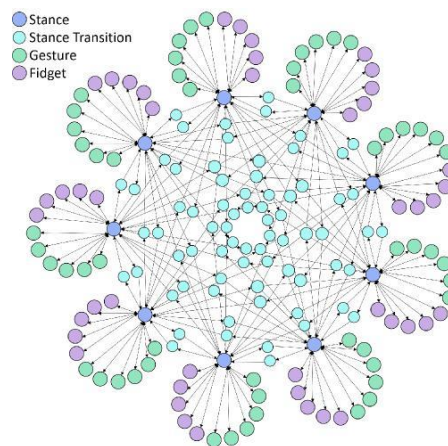


Figure 1. Explicitly creating the animation state machine with all connectivity information quickly becomes complex and prone to errors. Since the logic for all transitions is the same (up to some motion-specific parameters), there is a lot of redundant information, which we can reduce to a simple interface as shown in Figure 2

## 2. RELATED WORK

### 2.1 Agency, Planning and Architecture related to Embodiment

Over the past years there have been many works on turning text and other modalities into actions. The seminal work by Cassell et al. on text to motion [Cassell et al. 1998, Cassell et al. 1999, Cassell and Vilhjálmsón 1999], culminating in [Cassell et al. 2001], use natural language processing together with manually designed rules and heuristics to create a sequence of nonverbal behaviors from text. Similarly rule based approaches have been used to create talking head animations from chat text in online games [Vilhjálmsón 2004]. The realization of the motions is often performed for the whole body, without advanced additive composition [Thiebaut et al. 2008] and the interface for the action specification is often the Behavior Markup Language BML [Kopp et al. 2006]. Similar recent work use besides text also speech (audio) features [Marsella et al. 2013], and have been applied to a virtual therapist conducting a depression screening therapy [DeVault et al. 2014]. Other works on text to motion, build a probabilistic model of gestures for a specific speaker from annotated video [Neff et al. 2008].

To play animation clips, often state machines are used, which can conceptually be viewed as graphs. The works [Arikan and Forsyth 2002, Kovar et al. 2002] focuses on automatically building such a graph based on similarity between poses

in large data-sets. To match animations to action specifications more accurately in time, others have investigated optimal matching with dynamic programming [Stone et al. 2004, Bozkurt et al. 2016], which requires allowing re-timing of the clips via warping. In our experiments, we found that the warpings cause the resulting motion to look unnatural.

### 2.2 Engineering Feature-based Maps

When the motion can be generated from a set of features, or parameters known to the agent, such as the audio (the speech), or high-level parameters such as the gaze direction, or even a part of the motion specified otherwise, such as the head orientation for gaze, then a large amount of variations can be synthesized automatically, especially when the parameter and the mapping are continuous.

Lip sync is one case where audio is analyzed and corresponding lip shape parameters are computed over time [Digital 2017, Entertainment 2018]. In the case of the body, recent work has explored to map audio features to the trunk motion of the character— mapping the volume of the audio to the trunk and head arc [Sakai et al. 2016]. Adding arm gestures in an override fashion from the shoulder downwards, removing

essentially all spine motion from the arm's action, results in a motion that looks uncanny. Hence some map the arm motion back to the spine motion to bring back some of the missing dynamics, as described in chapter 9 of [Tanenbaum 2018].

## 2.3 Machine Learning

Mapping audio or speech to lip motion is a relatively well defined problem as strong correlations exist between the speech and mouth shape used to produce the sounds, resulting in many methods and papers on the topic, summarized in [Mattheyses and Verhelst 2015]. Researchers have pushed the envelope with speech-driven eyebrow motion [Ding et al. 2013], and more recently with speech-driven facial animation, which can produce motions in a given expression [Karras et al. 2017, Taylor et al. 2017, Sadoughi and Busso 2017]. In case of the body, experiments have been conducted to learn a mapping from speech to body motion, by first capturing the motion of an actor while talking [Levine et al. 2009, Levine et al. 2010]. Similarly an interesting recent work seeks to map music audio to body motion performed while playing the instrument to produce the sounds [Shlizerman et al. 2017].

But generally the results are poor, perhaps due to the articulated nature of the human body, but most likely due to the lack of powerful latent structures that can model the natural gestures that accompany speech content. Recent works experimented with the idea that additional modality, such as the face of the person talking to the avatar, together with audio and transcripts, could improve the results in a deep learning setting [Chu et al. 2018].

## 3. OVERVIEW

It is not possible to simply play any motion at any point in time when using the standard joint space interpolation provided in game engines such as Unity or Unreal Engine. For example, consider a gesture that moves the arm up, and then a sudden request for an action that starts with the arm pointing downward. Interpolating between both motions will cause a visual jump.

To avoid these artifacts, engineers and designers encode which motion can be played when or after which other motion as connections in a state machine, where the states are the motion clips. For example, consider arm motions and following our taxonomy guideline: stances for the arms such as hands on the hips, transitions between different stances, and all the gestures and fidgets that can be played from those stances. Imagine we have 9 stances, and about 5 gestures and fidgets on each stance. We would end up with a graph that looks like Figure 1. Authoring and maintaining these connections, requires exponential effort as the number of animations grows, and the probability of making mistakes gets increasingly high. In consequence very few embodiments scale up to rich and highly varied conversation behaviors. When looking at this large graph, we can observe the self-similarities between the sub-structures of the graph. We can generalize these sub-structures into meta nodes, as shown in Figure 2. This generalized graph is a perfect fit for a simplified drag-and-drop interface that automates the connectivity logic, while animators focus on the motions (Section 4).

To this point, we discussed arm motion, but when people are talking, they perform head motions such as nods, while doing different hand gestures and occasionally changing body postures such as shifting the weight to one side of the body. Creating all the combinations of head motions with gestures, fidgets and weight shifts leads to exponential content authoring. To cope with this complexity, we break down the motion space into body part layers (we call body, arms, head), and compose them in an additive fashion (Section 5). Note that our taxonomy still holds for the different body parts. Hence, the graph on each layer is the same, and we explain in Section 4 how the interface for specifying the motions on each layer works.

Finally, we tackle the problem that different agents can operate at different levels of abstraction, which makes it difficult for the embodiment to be used in practice. An agent might specify action types such as “beat” gesture at a given time, or have specific gestures it wants to play. Another agent might want to simply send dialogue lines without any actions. By modeling the probability distributions of the gestures, we designed a planner that can translate action sequences at different levels of abstraction into specific action sequences without conflicts (Section 6). In other words, our planner can generate plausible action sequences when none provided, and can avoid repetitive gestures when more abstract actions are specified. That said, we begin by describing the user interface for specifying the graph.

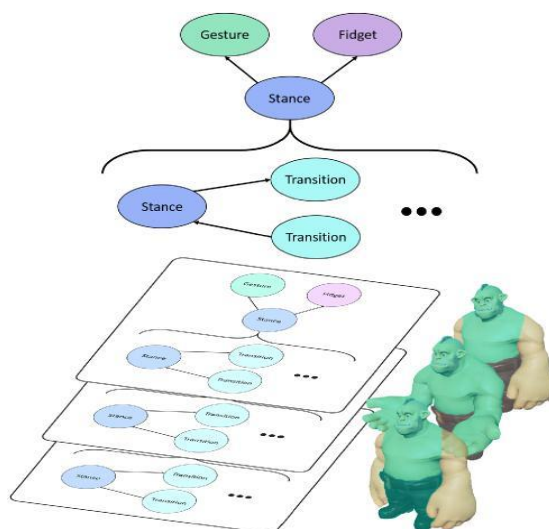


Figure 2. We can observe the self-similarities in the full graph (Figure 1), and generalize the connections between meta nodes corresponding to our taxonomy (stance, gestures, fidgets and stance transitions). This abstract graph, we call the meta-graph, naturally leads to a simplified interface for users to simply drag-and-drop animation clips over the nodes, as described in Section 4 and shown in Figure 3

## 4. USER INTERFACE

Based on the generalized meta graph shown in Figure 2 we devise a simple drag-and-drop interface to automate the connectivity logic between the animations as illustrated in Figure 3.

First the current stance, which is the central piece, is specified either by selecting an existing one or by drag-and-dropping a new one. Gestures and fidgets are then simply added by drag-and-dropping them onto the proper nodes. Similarly transitions are added, but first the other stance has to be selected. Any additional properties such as layer masks and timing together with optional details such as semantic information or base likelihood are specified in the properties panel on the right. By drag-and-dropping animations we fill the graph data structure, which is a multi-index map, where for each stance and meta-node type (taxonomy element type), we have a list of compatible motions. Additionally for each transition motion we also store in which stance it ends. Next we detail the inner workings of the state machine playback, together with the additive layer composition.

## 5. ANIMATION COMPOSITION

Creating all combinations of head motions with gestures, fidgets and weight shifts is not feasible and we therefore break the motion space down into body layers. Specifically we decompose the motions into three layers: body, arms, and head. Simply composing the layers by masking, results in robotic and uncanny motions, because the dynamics for other body parts is lost. Instead, by composing the motions for the different layers additively, we can maintain this dynamics. From analyzing the motions we observed the following influences between the three layers (body, arms, head) and the four body parts (legs, spine, arms and head):

*body* → *head, spine, legs*  
*arms* → *head, spine, arms*  
*head* → *head, spine*

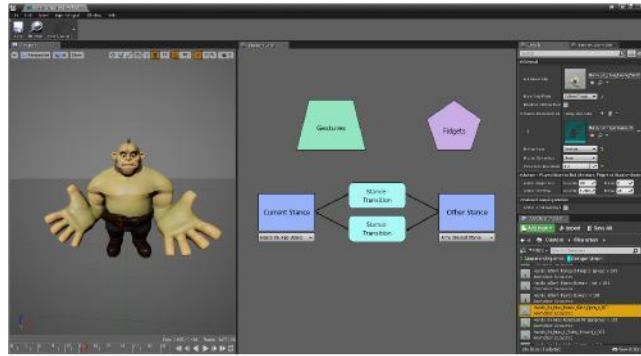


Figure 3. To add new motions, users drag-and-drop animation clips onto the corresponding meta nodes. Additional properties such as layer masks and timings can be specified in the properties panel

The final pose for each body part is thus composed as the additive combination of these three layers as visualized in Figure 4. The base pose used for this is a neutral pose, where the character has the hands at his side. To create an additive animation, all joint.

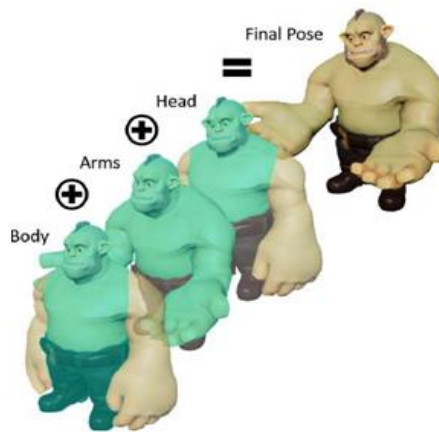


Figure 4. The animations triggered on the different layers head, arms, and body, are composed additively to build the final pose. The influences of the layers on the different body parts are highlighted in green

Transforms  $Q_{anim}$  are converted into offsets  $\tilde{Q}_{relative}$  to those of the base pose  $Q_{Base}$  such that  $Q_{anim} = Q_{Base} \otimes \tilde{Q}$  where  $\otimes$  is the additive operator (multiplication for orientations and addition for translations). The final poses for all layers are then composed as:

$$\begin{aligned} Q_{Head} &= Q_{Base} \otimes \omega_b \tilde{Q}_b \otimes \omega_a \tilde{Q}_a \otimes \omega_h \tilde{Q}_h, \\ Q_{Spine} &= Q_{Base} \otimes \omega_b \tilde{Q}_b \otimes \omega_a \tilde{Q}_a \otimes \omega_h \tilde{Q}_h, \\ Q_{Legs} &= Q_{Base} \otimes \omega_b \tilde{Q}_b, \\ Q_{Arms} &= Q_{Base} \otimes \omega_a \tilde{Q}_a, \end{aligned}$$

where  $\omega$  are influence weights,  $\tilde{Q}$  additive offsets,  $Q$  final poses and  $\{h, a, b\}$  refer to the *head*, *arms* and *body* layers. For the influence weights we used  $\omega = 1.0$  for all layers. Despite not normalizing the weights, we did not observe any visual artifacts. We assume that with more extreme motions, it might be necessary to perform some normalization, possibly through optimization.

A final note: to make the character more alive we apply a breathing motion that affects spine, head and arms (slight shoulder motion), lip-sync for the speech and life-like eye motion, further detailed in Appendix A. These are applied in the same additive fashion as the other motions. Now that we have a lively character, we look at how to combine it with the agents actions.



## 6. PLANNING AND MOTION SYNTHESIS

In practice it may be hard to predict at which level of abstraction an agent is going to operate: will it be at the level of "say line X", or will it be at the level of "lift right hand index finger by 10 degrees"? To support the various levels of abstraction, we designed a *planner* that first converts streams of actions from the agent, into *meta actions*, which are a unified representation containing all required meta data.

The meta actions (shown in Figure 5) contain an abstract action field with values that match our taxonomy: "gesture, fidget, stance transition", as well as a specific action field, together with an additional property field for dimensions such as positive - negative, and finally a timing field. This tuple of abstract and specific allows to accept both, more specific actions, as well as more abstract actions.

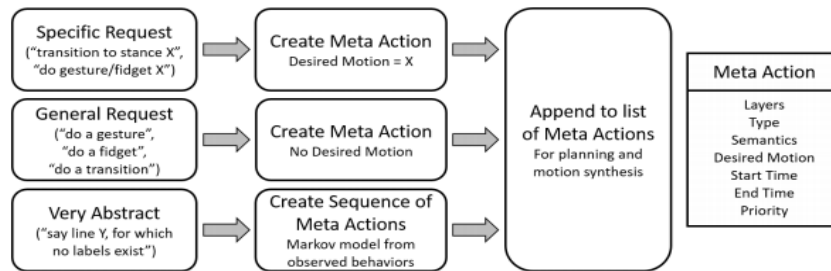


Figure 5. Action requests at various levels of abstraction are converted into meta actions, a unified representation containing all required data for further planning (Section 6.1) and motion synthesis (Section 6.2)

The next ingredient to our planner are probabilistic models of the gestures and actions. In the event of unusually absent actions, or when an agent acts only at a very high level such as "utter line X", the planner generates sequences of *meta actions* using a sequential probability distribution. We compute a Markov model from *labeled* real world casual conversations, which we detail further in Section 7.3.

Once we have the sequences of *meta actions*, our planner turns them into specific actions: first by resolving temporal conflicts, then by making sure gestures don't repeat using a probability distribution of the actions conditioned on past actions.

### 6.1 Temporal Replanning

Besides potential conflicts among the input meta actions, there may be conflicts with an already existing *meta plan*, which is a conflict-free sequence of meta actions as shown in Figure 6, created at an earlier timestep. From the existing plan we first recover all meta actions that start after the currently active action and merge them with the inputs. Through temporal replanning we then create a meta plan by adjusting the timings of the meta actions in a greedy way according to start time and action priority as illustrated in Figure 6.

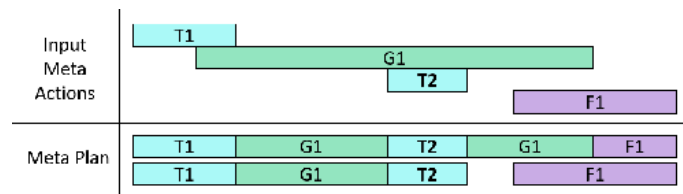


Figure 6. Through temporal replanning the meta actions are converted into a meta plan, a conflict-free sequence of meta actions, by adjusting the timings greedily according to start time and priority. For long actions interrupted by shorter actions of higher priority we either continue with the remaining part after the cut or discard it

### 6.2 Sampling Specific Actions

In the last stage, the meta plan is translated into a sequence of specific actions (animation clips). For a given meta action, we first retrieve all potential candidates by matching meta action properties against the specific

motion data set. The possible actions may introduce conflicts temporally, as the three cases shown in Figure 7. We remove these from the possible actions.

From the final set of possible actions, we sample according to a distribution that is conditioned on past actions. We assume here that all actions are uniformly distributed, unless specified differently through the interface, but modulate the distribution to reduce the probability of having recurrent motions. We also consider the three cases illustrated in Figure 7 in our sample selection.

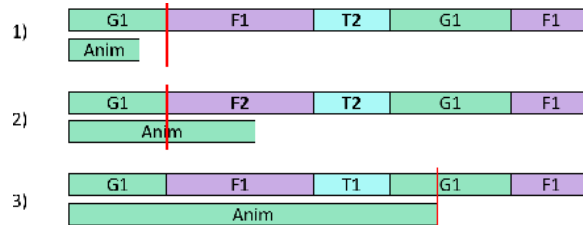


Figure 7. To synthesize a motion, animations are sampled according to the meta action properties. We distinguish three cases during this process: 1) If the animation ends before the meta action, we continue at the begin of the next meta action. 2) The animation has to end before a meta action with higher priority. If none is compatible, the meta action is discarded. 3) Meta actions completely overlapped by the animation are discarded and we continue at the end of the animation

Given a count of past actions  $\{c_i\}$  for the  $n$  candidates, the probability of picking candidate  $k$  is:

$$p_k = \frac{\frac{1}{\max(c_k, 1)}}{\sum_{i=1}^n \frac{1}{\max(c_k, 1)}}$$

When an animation has been used, its counter is updated as:

$$c_k = c_k + a,$$

where  $a$  is a weight determining how quickly the probability decreases the more often a motion is used. We used  $a = 4$ . Now that we have the agent connected to the animation system, we continue with experiments and results.

## 7. EXPERIMENTS AND RESULTS

One of the benefits of our taxonomy is to effectively guide animators and artists on which motions to create: stances, stance transitions, gestures and fidgets. However, it is quite challenging to evaluate these benefits, as creating the animations for an embodiment each time represents months of labor.

That said, we could leverage our taxonomy into automation, and the time required to populate the state machine with actions is much lower than with the standard state machine utilities in a game engine. In practice, the state machine increases in complexity as the number of motions increases, and requires engineering skills to find problems (see Section 7.1 for additional discussion on authoring state machines). To confirm this reality, we conducted an evaluation by measuring the time taken for an animator to populate the state machine using our interface, and the time taken using the standard state machine utilities in Unreal Engine.

Our animator has three years' experience with the Maya software, and has no engineering background. We first explained the interface with an example of 2 stances, 1 gesture, 1 fidget and 2 stance transitions, each explaining the layers for the body parts and motion-specific parameters. It required a total of 8 hours spread over two days (1.5 days of work) to We conducted several experiments to demonstrate the look and feel of our embodiment, which can be viewed in our accompanying video. However, we did additional work to accommodate agent actions at different levels of abstraction. To demonstrate this capability we conducted 3 experiments. First we hand-labeled videos of dyadic conversation with meta labels (see Section 6) and show that the embodiment can generate variations of the same meta action sequences (Section 7.2). A second experiment demonstrates the combination of two different streams of actions: a meta-actions stream together with specific actions coming from an interactive mimicry module (Section 7.4). Finally, we learned probability



distributions of the meta actions and show that we can generate plausible performances in the absence of actions from the agent (Section 7.3).

Populate the state machine with our 152 animation clips. Using directly Unreal Engine, our animator had a 1 week tutorial on authoring state machines with frequent help required to accomplish tasks. After this 1 week tutorial, our animator started creating a state machine manually. It took 63 hours (8.5 days) of work to finalize an equivalent state machine. The resulting motion holds several jumps and artifacts and it holds many more mistakes.

## 7.1 Discussion on State Machine Authoring

When authoring a state machine in Unity or Unreal Engine (or any modern game engine), motions are added by creating a new state. Several properties such as looping or not, start and end times, and so on, have to be specified. To use this state, all the connections to other states (clips) have to be established, which as the state machine grows, becomes more and more demanding. The growing number of connections shown on screen quickly leads to a visual overflow, making it highly prone to errors. For example, forgetting a connection or connecting to the wrong state leads to visual artefacts in the performance. Additionally, a transition requires transition logic such as the time at which it is possible to transition, and the type of blend between clips. This process is quite redundant as actions of the same meta class, all behave the same in terms of transitions. In contrast, our interface has transition logic pre-defined for the meta action classes: stance, stance transitions and the gesture/fidget categories.



Figure 8. Face-to-face conversation between a user and the virtual character. To generate the action sequences, we annotated audio as shown in Figure 10



Figure 9. Due to the probabilistic nature of the motion synthesis, for the same abstract action sequence different performances emerge. This includes using different animations as well as different timings

## 7.2 Annotating Dyadic Conversations (Video)

We video recorded two hours of dyadic conversation: 2 times 1 hour, each with 2 subjects. A list of casual topics was given to each subject in case conversation runs dry. We captured the upper body above the knees, as can be seen in our accompanying video and Figure 10.

To annotate the videos with our abstract action labels (stance, stance transition, gesture or fidget), we used the ELAN software [Max Planck Institute for Psycholinguistics 2018], shown in Figure 10. To capture the different combinations of actions, we created labels for multiple body layers: two layers for the head (one for stance and one for action such as nodding), one layer for each arm, and one layer for the body, legs and spine. For the arms, the labels included gesture, fidget and stance-transition. For the head action, the labels included nodding and shaking. And for the head stance and body layers, the labels only included stance-transition.

Given these sequences of abstract action labels, we can synthesize a performance by feeding them to our planner. Since the labels remain abstract, we can synthesize different performances for the same input sequence. Figure 9 shows an example of various hand gestures, synthesized from the same input.

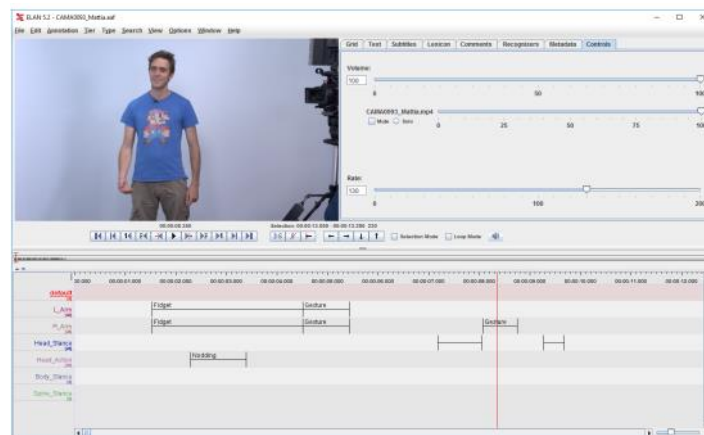


Figure 10. To create action sequences for casual conversations, we annotated the recorded dyadic conversations using the ELAN software [Max Planck Institute for Psycholinguistics 2018]. These annotations include abstract actions for different body part layers, which include head stance, head action, left arm, right arm and legs. These annotations can then be used either directly as actions (Figure 8), or to learn a Markov model (Section 7.3)

## 7.3 Modeling Action Distributions with a Markov Model

In the absence of labels, the embodiment can generate plausible conversation motion by leveraging the label distributions. We build a first-order Markov-model from the labeled video and sample the distribution to generate actions sequences. The possible states of the Markov model are the abstract actions gesture, fidget and stance transition. When additional semantic information is required such as positive for nodding, we can either randomly pick a semantic label when landing in a gesture state, based on its distribution in the annotations, or we can include it into the Markov model as additional states (gesture-without-semantic, positive-gesture, negative-gesture etc.). Then at generation time, we take the current Markov state and sample the next state based on the observed transition probabilities. Figure 11 shows a possible transition matrix for the different Markov states and Figure 12 shows different generated performances. The full performances can be seen in the accompanying video.

	L, G	L, F	L, T	R, G	R, F	R, T	LR, G	LR, F	LR, T
L, G	25%	37.5%	12.5%	12.5%	0%	0%	12.5%	0%	0%
L, F	26.7%	33.3%	6.7%	13.3%	20%	0%	0%	0%	0%
L, T	20%	20%	0%	0%	60%	0%	0%	0%	0%
R, G	0%	28.6%	0%	42.9%	0%	0%	14.3%	0%	14.3%
R, F	0%	42.9%	14.3%	0%	0%	0%	14.3%	14.3%	14.3%
R, T	0%	0%	100.0%	0%	0%	0%	0%	0%	0%
LR, G	25%	0%	0%	25%	25%	0%	0%	25.0%	0%
LR, F	0%	33.3%	33.3%	0%	0%	0%	33.3%	0%	0%
LR, T	0%	0%	0%	0%	20%	20%	0%	0%	60%

Figure 11. Using the labeled audio, we can build a first-order Markov-model. The transition matrix, in this example for the arms, shows the probability between the different states of the model, which consist of the left hand (L), right hand (R) and both hands (LR) together with the actions gesture (G), fidget (F) and transition (T)



Figure 12. Using the learned Markov model, we can generate new action sequences that look natural. Since the generated actions are still abstract, the resulting performances will vary

## 7.4 Video-based Mimicry

To demonstrate combining two streams of actions, one abstract and one specific, we took as input a mimicry module that reads the pose of a person in a video and classifies the stance, together with the stream of actions for the arm and head gestures. Given the detected stances, we pass an action of the form “transition to the detected stance” to the planner. If the stance is different from the current stance, the planner then finds the corresponding transition to trigger. Similarly the mimicry module also detects the orientation of the spine and head and then passes an action of the form “shift weight to the detected side” or “tilt head to the detected side”. Again, if the requested side for weight shift or head is different from the current, the planner finds the corresponding motion to trigger.

## 8. CONCLUSION

We introduced a compact action taxonomy for chit chat embodiment, together with a fast interface for labelling motion clips at a large scale (152 actions in our experiment). Users require no engineering skills to connect hundreds of motions, with regard to their body part and action type (stance, gesture, fidget and stance transition): they simply drag-and-drop clips onto the corresponding sheet and adjust timings in a side window. Compared to the standard state machine authoring system, our interface avoids many human mistakes and requires significantly less time (7 times faster by our measurement). Holding a large number of motions and variations yields richer and more natural looking performances, as can be seen in our accompanying video. Finally, our fast labeling interface could lead to additional automation: given a large set of labeled actions, we could investigate training a predictor to label new similar motions—allowing to fully automatically add motions to the agent’s embodiment.

## REFERENCES

- Arikan, O., 2002. Interactive motion generation from examples. *ACM Trans. Graph Publishers New York USA* pp. 483-490.
- Bozkurt, E., 2016. Multimodal analysis of speech and arm motion for prosody-driven synthesis of beat gestures. *Speech Commun. Vol. 85 C* pp. 29-42.
- Cassell, J., March 1999. Fully embodied conversational avatars: Making communicative behaviors autonomous. *Autonomous Agents and Multi-Agent Systems. KluwerAcademic Publishers Vol. 2, Issue 1*, pp. 45-64.
- Cassell, J., 1998. An architecture for embodied conversational characters. MIT Media Laboratory Publishers Cambridge Massachusetts USA.
- Bickmore, T. September 1999. Requirements for an architecture for embodied conversational characters. In *Computer Animation and Simulation, Eurographics Publishers Italy*, pp. 109-120.
- Cassell, J., January 2001 *Beat: The behavior expression animation toolkit. New York, USA*, pp. 477-486.
- Chu, H. 2018 *A face-to-face neural conversational model. Toronto, Canada*. pp. 7113-7121.
- Devault, D. January 2014 .*Simsensei kiosk: A virtual human interviewer for healthcare decision support. AMAS Paris France*. Pp. 1061-1068.
- Ding, Y. 2013. Speech - driven eyebrow motion synthesis with contextual markovian models. *Houston USA* pp. 3756-3760.
- Lehtinen, J. July 2017 *Audio-driven facial animation by joint end-to-end learning of pose and emotion. ACM Trans. Graph. Publisher New York USA. Vol. 36, No. 94*.
- Rick, J. 2007. *The behavior markup language. In Intelligent Virtual Agents, Springer Publisher, Berlin Germany*.
- Kovar, L. July 2002. *Motion graphs. ACM Trans. Graph Publisher New York USA. vol . 21*, pp. 473-482.
- Levine, S., December 2009 *Real-time prosody-driven synthesis of body language. ACM Trans. Graph Publisher California USA. vol. 28*, pp. 172.
- Levine, S. July 2010. *Gesture controllers. ACM Trans. Graph Publisher. vol. 29*, pp. 124.
- Marsell A. 2013. *Virtual character performance from speech. In Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation. New York USA. pp. 25-35*.
- Mattheyses, W. February 2015. *Audiovisual speech synthesis: An overview of the state-of-the-art. Ghent Belgium vol. 66*, pp. 182-217.
- Coutrot, A., February 2015 *Saccadic model of eye movements for free-viewing condition*.
- Hans, P. March 2008 *Seidel Gesture modeling and animation based on a probabilistic recreation of speaker style. ACM Trans. Graph. Publisher vol. 27 no. 5*.
- Sadoughi, N 2017. *Speech-driven animation with meaningful behaviors. Texas USA*
- Sakai, K. 2016. *Speech driven trunk motion generating system based on physical constraint. New York pp. 232-239*.
- Mattheur S. 2004. *Speaking with hands: Creating animated conversational characters from recordings of human performance. ACM Trans. Graph Publisher vol. 23 no. 3 pp. 506-513*.
- Tanenbaum, J 2014. *Nonverbal Communication in Virtual Worlds: Understanding and Designing Expressive Characters. ETC Press Publisher Pittsburgh USA*.
- Taylor S. July 2017. *A deep learning approach for generalized speech animation. ACM Trans. Graph Publisher Vol. 36 no. 4 pp. 93*.
- Thibeaux, M. 2008. *Smartbody: Behavior realization for embodied conversational agents. In Proceedings of the 7th International Joint Conference on Autonomous Agents and Mul-tiagent Systems – Estoril Portugal vol. 1 pp. 151-158*.
- Almsson, H. 2004. *Animating conversation in online games. In Entertainment Computing – Publisher Springer Berlin Germany pp. 139-150*.