Teaching Virtual Characters How to Use Body Language

Doron Friedman and Marco Gillies

Virtual Environment and Computer Graphics Lab,
Department of Computer Science,
University College of London
{d.friedman, m.gillies}@cs.ucl.ac.uk

Abstract. Non-verbal communication, or "body language", is a critical component in constructing believable virtual characters. Most often, body language is implemented by a set of ad-hoc rules. We propose a new method for authors to specify and refine their character's body-language responses. Using our method, the author watches the character acting in a situation, and provides simple feedback on-line. The character then learns to use its body language to maximize the rewards, based on a reinforcement learning algorithm.

1 Introduction

Social interaction is a core part of human life, and social behavior has become a key research area in Intelligent Virtual Agents (IVAs). Non-Verbal Communication (NVC) is vital to social interaction; it consists of all the signals sent between people that are not contained in language utterances. NVC is responsible for many aspects of social interaction: expressing emotion; regulating turn taking in conversation, and defining and expressing social relationships. These signals are often picked up subconsciously, without being explicitly noticed or understood. NVC varies greatly between people and can be an extremely useful method of making virtual agents that have their own individuality and personality. In addition, there is a large variation in NVC across cultures. For these reasons the generation of NVC has become one of the main challenges and one of the most active areas of IVA research. In this paper we suggest a new approach based on the ability of humans to judge NVC, and we present first results from our work in progress.

The typical approach for constructing characters with NVC relies on results from psychology, complemented by empirical data. This approach has been extremely fruitful, a prime example being the work of Cassell and her group [1,2], and was also used by one of the present authors [3,4]. However, there are limitations to this approach. First, our theoretical understanding of NVC is still incomplete. Second, and possibly more important, obtaining and analyzing data can be extremely time consuming and costly. We thus propose a new approach that tries to leverage the human capacity for evaluating NVC, without explicitly

T. Panayiotopoulos et al. (Eds.): IVA 2005, LNAI 3661, pp. 205–214, 2005.

being able to define it. This is done by allowing humans to watch a character act in a specific context and provide a simple feedback, every few seconds: whether the character's NVC is appropriate to the situation or not.

In this paper we discuss the first steps in this work in progress. First, we devised a method that combines exploration and generalization to allow the user to quickly prune and evaluate a large space of states and actions. Next, we have adapted a reinforcement learning (RL) [5,6] algorithm that allows the character to acquire a policy for making the right NVC actions to achieve a long-term reward.

Our focus in this paper is on the adaptation of RL to the domain of virtual characters with NVC-related behavior. We provide a description of our methods and the insights we have gained throughout their construction and initial evaluation.

2 Related Work

Machine learning has been adopted to train believable agents in virtual environments. Blumberg et al. [7] have trained an autonomous animated dog based on a real dog trianing technique called "clicker training". They demonstrate that the autonomous dogs can recognize and use acoustic patterns as cues for actions, as well as synthesize new actions from novel paths through its motion space. Isbell et al. [8] report on a software agent with RL capabilities inhabiting a multi-user text-based virtual environment. The agent was trained to proactively take actions in a social context by receiving rewards from other users in the VE. Conde, Tambellini, and Thalmann [9] demonstrate how agents can learn, using RL, to explore a virtual environment in an efficient yet flexible way.

Our research is aimed at using RL to learn non-verbal communication. There have been a number of general computational models of NVC. These include: Cassell *et al.*'s various systems, and particularly their virtual real-estate agent, Rea [1]. Guye-Vuillème *et al.* [10] have demonstrated avatars with a wide range of controllable expressive behavior. The Affective Presentation Markup Language (APML) is an XML-based language for defining the expressive behavior of characters [11].

Non-verbal behavior is generally divided into a number of modalities, many of which have been studied by virtual characters researchers. These include: facial expression(Pelachaud and Poggi [12]), eye gaze (Cassel et. al. [1], Rickel and Johnson [13], Garau *et al.* [14], and Gillies and Dodgson [3]), and style of motion (Chi et. al. [15]). In this paper we use the modalities of proxemics (or personal distance[1]), which has been little studied for virtual characters, in addition to posture and gesture.

Among research on posture, Cassell *et al.* [17] have investigated shifts of postures and their relationship to speech. Bécheiraz and Thalmann [18] use posture to display social closeness or distance between characters.

[1] The term "proxemics" was coined by the researcher E.T. Hall [16] when he investigated people's use of personal space.

The generation of gestures has been studied by a number of researchers. For example, Cassell *et al.* [1] have produced a character capable of extensive non-verbal behavior including sophisticated gestures. Chi *et al.* [15] present a way of generating expressive movements, similar to gestures using Laban notation. Gestures are closely related to speech and should be closely synchronized with it. Cassell, Vilhjálmsson, and Bickmore [2] present a system that parse text and suggests appropriate gestures to accompany it.

3 The Approach: Training Characters

NVC includes eye gaze, head motion, whole body movement, arm gestures, facial expressions, etc. A character acting in a virtual environment can have any combination (Cartesian product) of these elements in every moment. This allows for a rich possibility of expression. The question is: what combination of body-language elements should the character display at any given moment?

For simplicity we assume that the character's behavior occurs in discrete steps. In each such step the system needs to select a combination of body-language elements. This combination will typically be different from the previous one, so every step will typically involve a combination of basic animations. We will refer to such set of body-language elements, or to the set of basic animations, as an *action*; this use is also consistent with RL terminology.

Our experience indicates that the most difficult problem is context dependence, i.e., how to choose the right action *in a given situation*. Our idea is to allow the author of a virtual character to introduce the character into a situation, observe its behavior, and train it in real-time, by giving it a simple feedback. The author may provide feedback at any moment during watching the scenario. The feedback is a grade on a five point scale ranging from "very good" to "very bad". Figure 1 displays our simple interface and a scenario involving two characters.

We have by now implemented a basic system and have evaluated it with a simple scenario. Assume we want to train Alice who is in conversation with Bob[2]. Bob, whom we call the partner, uses NVC based on a pre-defined behavior mapping. Alice performs NVC behavior based on inputs from the learning component.

3.1 Animation Generation

We use the Demeanour architecture [19] to generate the non-verbal behavior of our characters. Demeanour is a general toolkit for creating character behavior. It is based around mappings from a number of inputs to output behaviors. A declarative language is provided for defining these mappings. In Bob's case the input is the state of the character and the output behavior consists of body movement (posture and gesture) and proxemics. The body movement engine uses a set of 33 pre-existing base motions (posture changes and gestures) to generate the behavior of the character. A number of these base motions are

[2] The conversation did not include actual speech.

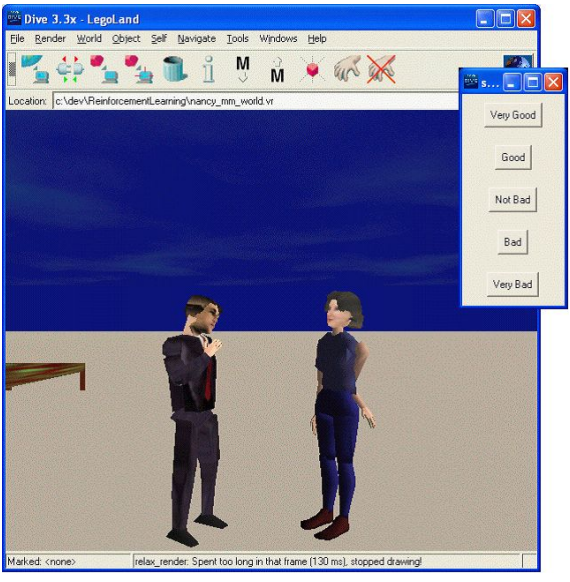

Fig. 1. A snapshot: the users watch the Dive window with the interacting characters, and are encouraged to hit one of the five rating buttons whenever they have an opinion on the recent NVC action performed by the female character

chosen based on the output of Demeanour and these are blended together to create a new motion. The proxemics engine works by choosing one of a set of simple motions (step forward/backard, turn left/right) to control the distance to another character to that character.

For the learning character, Demeanor divides the NVC components into five classes depending on the part of the body used and the type of motion:

Arm gestures: conversational (beat) gestures, crossing arms, scratching head
Body postures: e.g., leaning forward or backward, being hunched over
Head postures: holding the head high, low or to the side
Head gestures: nodding or shaking the head
Distance: moving forward or backward

At each time step, one of each class is chosen (an empty motion can be chosen for each class), and the resulting animation is performed simultaneously.

Demeanor was implemented on top of Dive [20,21]. The RL component was implemented in Matlab[3]; this allows rapid prototyping. Our configuration allows running the animation and the learning components on different machines, which communicate using the network protocol VRPN[4]. This, together with the fact that we use Dive, will allow us to evaluate the system in immersive virtual reality (VR), and in multi-user settings.

[3] `http://www.mathworks.com`
[4] `http://www.cs.unc.edu/Research/vrpn`

3.2 Defining the Learning Problem

In RL, problems of decision-making by agents interacting with uncertain environments are usually modelled as Markov decision processes (MDPs). In the MDP framework, at each time step the agent senses the state of the environment, and chooses and executes an action from the set of actions available to it in that state. The agent's action (and perhaps other uncontrolled external events) cause a stochastic change in the state of the environment. The agent receives a scalar reward from the environment. The agent's goal is to choose actions so as to maximize the expected sum of rewards over some time horizon. An optimal policy is a mapping from states to actions that achieves the agent's goal.

In our case, the state space is comprised of a combination of four factors that we call *classes*:

1. partner conversation state — speaking, listening, or none,
2. partner mood — neutral or unhappy,
3. learning character conversation state — speaking, listening, or none; and
4. proximity — five categories from very near to very far.

If the classes are denoted by C_1, C_2, C_3, and C_4, then each state is a tuple $< c_1, c_2, c_3, c_4 >$ such that $c_1 \in C_1$, $c_2 \in C_2$, $c_3 \in C_3$, and $c_4 \in C_4$.

The actions are similarly arranged in five classes, which correspond to the animations as described in Section 3.1. The reward, in our case, is an integer number between 1 (negative reward) and 5 (positive reward).

We have started with a simple approach for quickly exploring the space, and have then extended it to use policy learning; the next two sections describe these two approaches.

3.3 Exploration with Generalization

First, we recognize that we rely on human feedback to learn the space, and in such context the space is quite large.[5] We want to quickly explore it and find the right actions, or the right NVC behavior, for every state.

In the first instance we are only concerned with immediate reward, and we assume a stationary environment. Thus, we assume there is an optimal value function that assigns a value to each state-action pair, $Q :: S \times A \rightarrow R$, where S is the set of all states, A is the set of all actions, and R denotes the real numbers.

Fortunately, our space is a combinatoric product of what we have called classes. Given a reward for an action, which is a combination of gestures and postures in different classes, we want to reward each class accordingly. This is sometimes referred to as the structural credit-assignment problem.

Our method is based on two principles. Given the n-dimensional space of state and action classes[6], we want to sample it in a smart way, and then generalize from our samples, using assumptions from our domain.

[5] The number of states is 120, the number of actions is 4275, so the space is comprised of 513,000 values.

[6] In our case $n = 9$.

Sampling may be done in several ways. A greedy approach would always pick the action with the highest Q value for the current state. In order to encourage some exploration of the space, it is possible to use an ε-greedy approach; this selects the optimal action with probability $1 - \varepsilon$ and a random action with probability ε. The parameter ε can be tuned to balance exploration versus exploitation. We have also tested Boltzmann exploration [5], which is popular in the context of RL. Using simulations we have found the ε-greedy exploration to work best, when ε is small and is gradually decreased. Since, in this stage, we are only interested in exploring the space, we do not repeat a combination of space-action more than once.

The second principle involves generalizing from the samples. The underlying assumption is that we can generalize a specific instance by extrapolating into the different dimensions of the space.

Typically, RL uses function approximation for generalization. This is necessary in the case of continuous or very large spaces. In our case we want to generalize because the space is large relative to the sample size, but it is discrete, and is not large in terms of the number of computations required. In addition, we want to take advantage of our knowledge about the combinatoric nature of the space. Note that we cannot assume that the space is continuous; our generalization principle is weaker. This will become clear below as we explain our method for generalization.

We define a and a' to be similar if a and a' have equal value for at least four out of five classes, or, more generally, we say that a and a' are $k - similar$ if they are equal in at least $n - k$ out of the total n number of classes.

For generalization, we assume that similar actions have similar value, or that there is some small enough K for which:

if a and a' are similar then $\forall s \in S : |Q(s, a) - Q(s, a')| < K$

We expect that this correlation might have exceptions, but we use it as a heuristic to try to find good candidate actions, which will then be rated by the user.

The same similarity heuristic holds for states. We define s and s' to be $k - similar$ if they are equal in at least $n - k$ out of the total n number of classes.

if s and s' are similar then $\forall a \in A : |Q(s, a) - Q(s', a)| < K$

Generalizing for states, in our domain, is more risky than generalizing for actions. For example, if the character is required to be submissive we would prefer her head to be low, regardless of whether she is speaking or listening. However, it is quite possible that we would want our character to respond to a happy partner in a very different way than to an unhappy character.

Such distinctions are much easier handled by knowledge-based approaches than by statical approaches such as RL. Although it is not impossible to in-

troduce such domain knowledge into the RL algorithm, we have used an easier resort: we assigned a smaller weight to state similarity.

Our algorithm is as follows. First we initialize $Q(s,a)$ to be the average reward for all s,a. The training includes a continuous iteration where we sample actions as explained earlier. The parameter ε is slowly decreased to slightly reduce exploration. Q is updated as follows: If the user provides a reward r to action a when in state s, then:

$$Q(s,a) = r$$
$$\forall s' \text{ similar to } s \text{ and } a' \text{ similar to } a : Q(s',a') = Q(s',a') + \alpha[r - Q(s',a')]$$

The constant α determines the rate of generalization[7]. In our case, we found that the update described above, together with a $2-similar$ generalization for actions only, result in covering 0.5% of the space per each user rate. This is enough to ensure learning with a reasonable feedback of a few hundred feedback data points. We can terminate the learning process when most of the space is covered.[8]

By the end of the training phase, we are left with a table that is assumed to be a good approximation of the value function Q. Based on this table a policy may be defined and used in real-time to drive the virtual characters. Such a policy would pick up, for each state, from the actions with a relatively high Q value.

3.4 Learning a Behavior Policy

In our initial evaluation of the exploration and generalization method we have found that it is possible to quickly learn that some gestures are better than others, in a given situation. However, we realized one of the limitations of this method: very often the facts that the agent needs to learn are reflected in the states rather than in the actions.

The main reason that this happens is that our domain requires learning with delayed reward. For example, say we want to encourage Alice to keep her head low. Assume Alice lowered down her head, and kept it low for a while. In this case she will probably get a relatively high reward for the whole duration that her head was lowered, and not only for the action that included lowering the head. Another example involves encouraging Alice to get farther away from the partner. Getting farther away might result in a large reward when she is far away, rather than an immediate reward for every backwards step.

This leads us to the more general problem involving IVAs that need to learn a behavior policy with delayed rewards. While some mappings can be learned using traditional supervised learning, such as immediate rewards and state transitions,

[7] We actually use different values for state generalization and for action generalization; since we want to be more careful with state generalization we use smaller values of α.

[8] If the space has larger dimensionality, we can update for $k-similar$ actions and states with higher k values.

policies cannot be learned this way. The RL framework is specifically intended for learning good policies in such conditions. While feedback is given by the user to the last state-action, it is "backed-up" to other state-action pairs by the RL algorithm.

RL may be best regarded as a framework of problems and approaches, and includes many specific techniques. We have selected Sarsa [5,6], which allows learning directly from raw experience, without having a model of the environment's state dynamics.

In this section we are interested in learning a policy: $\Pi :: S \times A \rightarrow [0, 1]$, i.e., we want to learn to choose the right actions in a given state with higher probabilities. In Sarsa we look at quintuple of events $(s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1})$, which makes up a transition from one state-action pair to the next. The update rule is:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_t + 1 + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

We use generalization, similar to the exploration and generalization as explained in Section 3.3. For the generalization to similar states and actions, we use the same formula, but instead of r_t we use $\frac{r_t}{k}$ where $k > 1$ reflects the rate of generalization.

We terminate the training phase when the user is satisfied with the learned policy. This policy may then be used in real time to control the character's behavior.

4 Discussion and Future Work

Using simulations we have validated the methods and found out the optimal values for the learning parameters. At this stage we are evaluating our approach using empirical experiments; we let users train characters and evaluate their perceived NVC capabilities, as compared with characters with random NVC, and with characters with hard-coded NVC behavior. This will eventually be done in a highly-immersive Cave-like [22] system, as part of our research on the sense of presence [23]. The results will be reported in a full paper.

Body language is, we believe, a good starting point to evaluate our method. Our animation platform includes other aspects of body language, which we intend to incorporate into the framework described here; this includes gaze direction and facial expressions. A more ambitious extension that we hope to explore is learning blending parameters for fine tuning of motions. This will entail learning in a continuous space, rather than a discrete combinatoric space.

Clearly, we need to train the characters and evaluate our system in the context of more complex scenarios. We see this as gradually leading to the construction of tools for authoring characters for non-linear narratives. Eventually, our method needs to be evaluated in the context of a complete application. This means we will need to extend our method to much larger state-action spaces, which means that our methods will need to be refined. Specifically, we intend

to further investigate our generalization principle and base it on more formal grounds.

While our work is in its early stages we can already draw some conclusions. We believe the general approach, that of letting humans provide high-level feedback to train character NVC, to be promising. We believe our approach as described here can be extended to cover a wide variety of situations and scenarios.

A second conclusion is that standard machine-learning algorithms need to be carefully adapted to the problem. We explained why purely symbolic approaches are not adequate for NVC, but we have still learned that domain knowledge should be integrated into the algorithm. This calls for the development of specialized algorithms, and for an approach that relies more heavily on empirical evaluation. In general, there is interest in the RL research community in knowledge-based methods for using domain knowledge in the learning process; such research needs to be adapted to our domain.

Acknowledgements

This work has been supported by the European Union FET project PRESENCIA, IST-2001-37927. The development of the Demeanour behavior engine was supported by BT, plc. We would also like to thank Mel Slater and the UCL Department of Computer Science Virtual Environments and Graphics group for their help and support.

References

1. Cassell, J., Bickmore, T., Campbell, L., Chang, K., Vilhjálmsson, H., Yan, H.: Embodiment in conversational interfaces: Rea. In: ACM SIGCHI, ACM Press (1999) 520–527
2. Cassell, J., Vilhjálmsson, H.H., Bickmore, T.: BEAT: the behavior expression animation toolkit. In: ACM SIGGRAPH. (2001) 477–486
3. Gillies, M., Dodgson, N.: Eye movements and attention for behavioural animation. Journal of Visualization and Computer Animation **13** (2002) 287–300
4. Gillies, M., Ballin, D.: A model of interpersonal attitude and posture generation. In Rist, T., Aylett, R., Ballin, D., Rickel, J., eds.: Fourth Workshop on Intelligent Virtual Agents, Kloster Irsee, Germany (2003)
5. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA (1998)
6. Kaelbling, L.P., Littman, M.L., Moore, A.W.: Reinforcement learning: A survey. Journal of Artificial Intelligence Research **4** (1996) 237–285
7. Blumberg, B., Downie, M., Ivanov, Y., Berlin, M., Johnson, M.P., Tomlinson, B.: Integrated learning for interactive synthetic characters. In: SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques, New York, NY, USA, ACM Press (2002) 417–426
8. Isbell, C., Shelton, C.R., Kearns, M., Singh, S., Stone, P.: A social reinforcement learning agent. In: AGENTS '01: Proceedings of the fifth international conference on Autonomous agents, New York, NY, USA, ACM Press (2001) 377–384

9. Conde, T., Tambellini, W., Thalmann, D.: Behavioral animation of autonomous virtual agents helped by reinforcement learning. In Rist, T., Aylett, R., Ballin, D., Rickel, J., eds.: 4th International Workshop on Intelligent Virtual Agents (IVA'03). Volume 272., Springer-Verlag: Berlin (2003) 175–180
10. Guye-Vuilléme, A., T.K.Capin, I.S.Pandzic, Magnenat-Thalmann, N., D.Thalmann: Non-verbal communication interface for collaborative virtual environments. The Virtual Reality Journal **4** (1999) 49–59
11. DeCarolis, B., Pelachaud, C., Poggi, I., Steedman, M.: APML, a markup language for believable behaviour generation. In Prendiger, H., Ishizuka, M., eds.: Life-like characters: tools, affective functions and applications. Springer (2004) 65–87
12. Pelachaud, C., Poggi, I.: Subtleties of facial expressions in embodied agents. Journal of Visualization and Computer Animation. **13** (2002) 287–300
13. Rickel, J., Johnson, W.L.: Animated agents for procedural training in virtual reality: Perception, cognition, and motor control. Applied Artificial Intelligence **13** (1999) 343–382
14. Garau, M., Slater, M., Bee, S., Sasse, M.A.: The impact of eye gaze on communication using humaniod avatars. In: ACM SIGCHI. (2001) 309–316
15. Chi, D., Costa, M., Zhao, L., Badler, N.: The emote model for effort and shape. In: ACM SIGGRAPH, ACM Press/Addison-Wesley Publishing Co. (2000) 173–182
16. Hall, E.T.: The Hidden Dimension. New York: Doubleday (1966)
17. Cassell, J., Nakano, Y., Bickmore, T., Sidner, C., Rich, C.: Annotating and generating posture from discourse structure in embodied conversational agents. In: Workshop on Representing, Annotating, and Evaluating Non-Verbal and Verbal Communicative Acts to Achieve Contextual Embodied Agents, Autonomous Agents 2001 Conference, Montreal, Canada (2001)
18. Bécheiraz, P., Thalmann, D.: A model of nonverbal communication and interpersonal relationship between virtual actors. In: Proceedings of the Computer Animation '96, IEEE Computer Society Press (1996) 58–67
19. Gillies, M., Ballin, D.: Integrating autonomous behavior and user control for believable agents. In: Third international joint conference on Autonomous Agents and Multi-Agent Systems, Columbia University, New York City (2004)
20. Steed, A., Mortensen, J., Frecon, E.: Spelunking: Experiences using the DIVE System on CAVE-like Platforms. In: Immersive Projection Technologies and Virtual Environments. Volume 2. Springer-Verlag/Wien (2001) 153–164
21. Frecon, E., Smith, G., Steed, A., Stenius, M., Stahl, O.: An overview of the COVEN platform. Presence: Teleoperators and Virtual Environments **10** (2001) 109–127
22. Cruz-Neira, C., Sandin, D.J., DeFanti, T.A., Kenyon, R.V., Hart, J.C.: The CAVE: Audio visual experience automatic virtual environment. Comm. ACM **35** (1992) 65–72
23. Slater, M., Usoh, M.: Presence in immersive virtual environments. In: Proc. IEEE Virtual Reality 1993, Seattle, WA (1993) 33–40