Data-Driven Programming and Behavior for Autonomous Virtual Characters

Jonathan Dinerstein, Parris K. Egbert, Dan Ventura and **Michael Goodrich**
Computer Science Department
Brigham Young University
jon.dinerstein@lmco.com, {egbert, ventura, mike}@cs.byu.edu

In the creation of autonomous virtual characters, two levels of autonomy are common. They are often called *motion synthesis* (low-level autonomy) and *behavior synthesis* (high-level autonomy), where an action (i.e. motion) achieves a short-term goal and a behavior is a sequence of actions that achieves a long-term goal. There exists a rich literature addressing many aspects of this general problem (and it is discussed in the full paper). In this paper we present a novel technique for behavior (high-level) autonomy and utilize existing motion synthesis techniques.

Creating an autonomous virtual character with behavior synthesis abilities frequently includes three stages: forming a model used to generate decisions, running the model to select a behavior to perform given the conditions in the environment, and then carrying out the chosen behavior (translating it into low-level synthesized or explicit motions). For this process to be useful it must *efficiently* produce *realistic* behaviors. We address both of these requirements with a novel technique for creating cognitive models. The technique uses programming-by-demonstration to address the first requirement, and uses data-driven behavior synthesis to address the second. Demonstrated human behavior is recorded as sequences of abstract actions, the sequences are segmented and organized into a searchable data structure, and then behavior segments are selected by determining how well they accomplish the character's long-term goal (see Fig. 1). The resulting model allows a character to engage in a very large variety of high-level behaviors.

Overview and Formulation

The proposed data-driven technique contains two interrelated parts: training via programming-by-demonstration and autonomous behavior synthesis. The training phase consists of 1) a programmer integrating our system into a new character, 2) a user demonstrating behaviors, and 3) the demonstrator testing and editing the behaviors. The synthesis is effected by 1) segmenting and clustering the demonstrated behaviors, 2) performing segment selection to choose the next behavior to perform, and 3) iteratively performing the actions in the selected segments. These processes are formalized in the full paper. Briefly, we assume a state/action representation and a nondeterministic environment which probabilistically maps state/action pairs to new states, $\mathbf{s}_{t+1} \sim Pr(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$.

Behaviors are represented as an ordered sequence of actions, $b = \langle \mathbf{a}_1,\ \mathbf{a}_2,\ ...,\ \mathbf{a}_l \rangle$, and are recorded by observing the user at discrete time intervals. Behaviors are segmented into a set Q of small, fixed length sequences and novel behavior is produced by concatenating behavior segments from Q. Behavior segments are selected such that their concatenation fulfills the goals of the animator and appears stylized and natural. Goals are encoded as a fitness function, and the sequence of states induced by a behavior (sequence of actions) can be evaluated against this (finite-horizon) fitness:

$$b^{best} = \underset{b_i(j, j+K_{length}) \in Q}{\text{argmax}} \left(\sum_{u=1}^{K_{length}} f(\tilde{\mathbf{s}}_u) \right)$$

where $b_i(j, j + K_{length})$ is the sequence of actions tested, $\tilde{\mathbf{s}}_1 ... \tilde{\mathbf{s}}_{K_{length}}$ are the states resulting from performing these actions, and $f : S \rightarrow \mathbb{R}$ is the fitness function.

Autonomous Behavior

Our approach makes deliberative decisions by simulating the outcome of candidate choices. Such simulation is analogous to traditional planning, except that we only consider a set of $n = |Q|$ paths in the search tree. In other words, each behavior segment $b_i \in Q$ represents a path through the search tree (and thus the user demonstration has, in essence, pruned the search tree, allowing faster deliberation). We further reduce computational requirements by hierarchically clustering behavior segments and using a greedy search that relies on the notion of a *prototype* member of a cluster. The fitness of a cluster is approximated by computing the fitness of its prototype segment. The affect of other agents in the environment is included in the simulation using their existing behavioral models, if known, or by using agent modelling techniques. Finally, since our technique is a heuristic one and since, more importantly, the environment is dynamic, it is important occasionally to reevaluate and perhaps modify behavior. During execution, the model periodically evaluates whether the currently selected behavior segment is still appropriate by determining whether the average fitness of the remaining actions is above a threshold. If not, the current segment is deemed invalid and a new segment is selected.

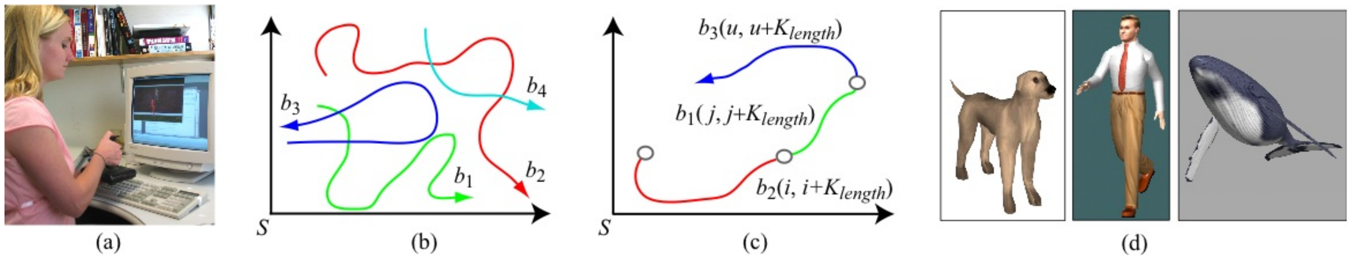

Figure 1: *Overview.* (a) A user interactively demonstrates the desired high-level character behavior. (b) These demonstrations are recorded as sequences of abstract actions. (c) At run-time, segments of the action sequences are combined into novel behaviors. The resulting sequence of abstract actions determines the categories and goals of the character's motion synthesis. Thus a character can engage in very long-term planning to achieve a virtual environment state $S(\mathbf{x})$.

Experimental Results

For this work, the most important evaluation criterion is the aesthetic value of the resulting animation. Since aesthetic value is subjective, the case studies and resulting video serve as the basis for this evaluation. Other criteria are also relevant, however, including the following: objective quality of action selection, sensitivity to parameter settings, algorithmic complexity, search time, memory requirements, CPU usage, and time required by the animator. We have performed five case studies and demonstrations in the experiments: a submarine piloting task, a predator/prey scenario, a crowd behavior scenario, a capture-the-flag game, and a cooperative assembly task (Fig. 2). In each case study, there is no explicit communication between the characters. All characters must rely on "visual" perceptions to ascertain the current state of the virtual world $\mathbf{s}_t$ and perception is performed by each character individually. The first three case studies use actions of short duration ($<$ 1 second). The last two use actions of long duration ($>$ 5 seconds). In all case studies, significant action planning is performed (up to 20 actions into the future).

Our programming by demonstration approach has empirically proven to take very little time and storage, with a small amount of behavior trajectory data sufficient to achieve effective character behavior, and an informal user case study provides evidence that our interface is intuitive and easy to use. Our technique provides deliberative decision making while being significantly faster and more scalable than a cognitive model because the use of example action sequences allows efficient computation in large action spaces. (See video at $rivit.cs.byu.edu/a3dg/videos/dinerstein\_ddpb.mov$.)

Our technique produces stylized and natural high-level character behavior that can be quickly demonstrated by non-technical users. While our approach is more computationally expensive than most reactive approaches to decision making, it can be argued that deliberative techniques (like ours) can produce superior behavior. Moreover, our technique, being O(log n), where n is the number of demonstrated action sequences, is significantly faster than traditional deliberative methods, which usually are O($m^{|A|}$), where $|A|$ is the number of possible actions and m is the depth of the planning tree. Because our technique is heuristic, optimal decision making is not guaranteed. However, we empirically demonstrate that the approach is robust, is effective in complex virtual environments, and is less computationally expensive than many traditional deliberative techniques. It is a surprisingly scalable deliberative scheme, and it can work with actions of fine or coarse granularity as demonstrated in our case studies. We can compute long plans (e.g. 20 actions), allowing us to be confident in these plans. Our technique works in conjunction with standard character animation algorithms, providing goals and guidance through long-term deliberation.

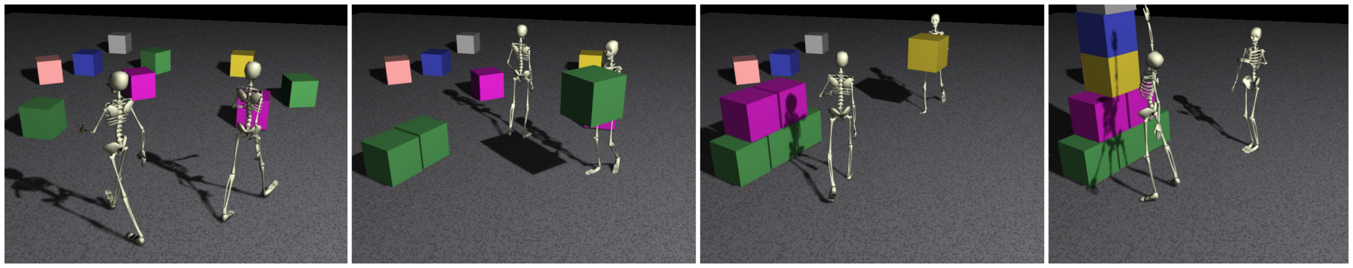

Figure 2: *Animation of the Assembly test bed.* Two characters work together, without explicit communication, to assemble a color-coded pyramid from boxes. The fitness function (which defines the desired shape and color configuration) is automatically learned during the demonstration phase. The characters choose high-level actions (e.g., "pick up a red box and place at location x"), which are translated into novel motion through a combination of data-driven motion synthesis and inverse kinematics.