Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2008)
M. Gross and D. James (Editors)

Puppet Master: Designing Reactive Character Behavior by Demonstration

James E. Young[1,3] and Takeo Igarashi[2,3] and Ehud Sharlin[1]

[1]University of Calgary [2]The University of Tokyo [3]JST ERATO

Abstract

*Puppet Master is a system that enables designers to rapidly create interactive and autonomous animated character behaviors that react to a main character controlled by an end-user. The behavior is designed by demonstration, allowing non-technical artists to intuitively design the style, personality, and emotion of the character, traits which are very difficult to design using conventional programming. During training, designers demonstrate paired behavior between the main and reacting characters. During run time, the end user controls the main character and the system synthesizes the motion of the reacting character using the given training data. The algorithm is an extension of Image Analogies [HJO*01], modified to synthesize dynamic character behavior instead of an image. We introduce non-trivial extensions to the algorithm such as our selection of features, dynamic balancing between similarity metrics, and separate treatment of path trajectory and high-frequency motion texture. We implemented a prototype system using physical pucks tracked by a motion-capture system and conducted a user study demonstrating that novice users can easily and successfully design character personality and emotion using our system and that the resulting behaviors are meaningful and engaging.*

Categories and Subject Descriptors (according to ACM CCS): I.3.6 [Computer Graphics]: Methodology and Techniques–Interaction Techniques

1. Introduction

Characters, such as those in animated films or computer games, or even autonomous robots interacting in the real world, are becoming increasingly common in everyday life. Having convincing, believable personalities and behaviors is very important for these characters, as it strengthens communication and suspension of disbelief, and ultimately results in a more rewarding, engaging, and comfortable experience [Bat94, Bre02, RN96]. In particular, it is critically important that interactive characters react convincingly to real-time user input while maintaining a coherent personality. For example, an aggressive merchant in a video game may chase after the user's character, a shy cleaning robot may hide from humans while cleaning, and a pet robot dog may jump happily when its owner returns home.

Programming a character's real-time interactive behavior is a difficult problem, particularly when trying to achieve a certain personality or interaction style. The logical, step-by-step state-machine style endorsed by conventional programming languages is good for specifying goal-oriented actions, but is not directly well-suited to the design of human-like traits such as personality and emotion; consider how difficult it is to program interactive shyness or anger using a series of logical states, rules and steps. Currently, programmers have to implement complex models to achieve believable results.

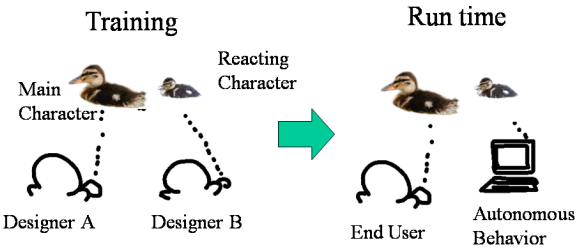


Figure 1: *First the behavior of a reactor is demonstrated in response to a main character's behavior. At run time, the reacting character's behavior is synthesized, reproducing the personality and emotion demonstrated in the training.*

Artists, however, such as actors, puppeteers, and so forth, have an incredible ability to develop interactive characters and personalities in various performance-based mediums such as puppet shows, plays, computer avatars and even remote-controlled robots. Unfortunately, conventional tools available to create autonomous interactive behaviors are often inaccessible to non-technical artists, in part because of their algorithmic nature and their incapability to explicitly express character personality and emotion. The result is that interactive behaviors are generated primarily by computer software engineers using logic-based algorithms, with results often being predictable and boring.

In this paper, we introduce a programming-by-demonstration approach to make the design of interactive character behavior accessible to artists. In the training phase a user (or two working collaboratively) demonstrates paired motion of two characters, with one character reacting to the actions of the other. At run-time, the end-user controls the main character and the system generates, in real time, the reactive behavior of the other character with the characteristics observed in the training data (Figure 1). Demonstrating an interactive behavior allows artists to encapsulate personality and emotion that they may not be able to logically explain using computer algorithms. Furthermore, training is quick (average 33s in our study) and generation is done in real time without preprocessing.

Our behavior-synthesis algorithm is an extension of the image analogies algorithm [HJO*01], which learns static image filters from example image pairs and applies them to a new input image. Similarly, our system learns reactive behavior from an example pair of motions and applies it to a new input motion. Our paper describes how we extend the original method to work for real-time, dynamic, reactionary locomotion behavior. Specifically, we introduce meaningful behavior-related features, a method for balancing between the similarity and coherence metrics, and we also separately synthesize general motion trajectory and motion texture, integrating them during the final stages of motion synthesis.

We built two prototype systems, one using a standard mouse as input and the other using a tabletop system with physical pucks tracked by a motion capture system. The tabletop system allows for simultaneous control of both the main and reacting characters, and allows the user to control characters' orientations. We ran a user study using the tabletop system asking one group of participants to design a set of character behaviors and the other group to interact with a set of designed behaviors. The results show that novice users can design interactive behaviors quickly using our system and successfully convey personality and emotion in the form of interactive behavior.

2. Related Work

There has been a great deal of work that aims for life-like, convincing interactive behavior. A common approach is to explicitly program the behavior model [BG95, Mae95, Rey87], where the programmer explicitly defines what to do for particular input scenarios. These systems require an understanding of the underlying algorithm and so are less accessible to the general artist, and do not support direct and intuitive design of emotion and personality.

Programming by demonstration was originally used to automate GUI operations [Cyp, MWK89, PG96], e.g., Pavlov [Wol97], explicitly for interactive agents, defines the low-level stimulus-response behavior of the agent using logical event sequences and conditionals. Similar work by Dinnerstein et. al [DE05, DEV] focuses on collisions, planning and goals, and requires the presence of a programmer. While successful, these systems do not provide tools to represent personality and emotion in the same way we do. Several systems design animation by performance demonstration [DYP03, HOCS02, IMH05b, IMH05a, TBvdP04] or apply the idea to robotic motion [FSMI00, RPI04]. These, however, focus on the playback of the demonstration and do not respond intelligently to user interaction.

Other systems focus on synthesizing new motion from an often large, pre-processed example database in real time [LL04, LCL07, WH97]. While some systems interactively respond to user input (joystick control, moving obstacles, other characters, etc), the mapping from the user input to the output is explicitly (and often tediously) defined by the programmer. Furthermore, the target of these systems (e.g., [HGP04]) is primarily the physical accuracy (punch, jump, walk, collision avoidance, etc.), not the explicit design of personality emerging from interactive motion.

Research in human-robot interaction is shifting from viewing robots as tools to designing affective robotic interfaces [Bre02, Nor04]. Robots are active participants in users' physical environments, and so the design of their form, posture and movement play a dramatic role in the quality of the resulting interaction [MMMI05]. Designing robotic actions and movement by demonstration has been investigated extensively (for example, [Bre02, FSMI00, RPI04, MMMI05]) with efforts ranging from simple movement playbacks to complex integrated actions. However, these methods target goal-oriented design of robotic pose and motion, and do not explicitly handle emotion or personality. Finally, it has been shown that observers attribute emotion to simple motions and actions [ABC, Kas82]. In our work, we enable designers to control (through demonstration) what sorts of motions and actions will be perceived.

3. System Overview

The ultimate goal is to allow the intuitive design of all aspects of behavior (gestures, facial expressions, eye movement, pose, etc) to create believable whole-body characters. As initial exploration this paper focuses on character locomotion (movement path). We implemented two interfaces

that enable designers to demonstrate behaviors: a mouse-based GUI and a tabletop Tangible User Interface (TUI).

3.1. Mouse-Based GUI

The mouse-based GUI (Figure 2) works on any regular PC. The standard mouse, however, lacks a rotation sensor and is designed for use by a single-user. This means that the main character and reactor behaviors must be trained sequentially and that the direction a character is looking cannot be explicitly specified, e.g., a character cannot move sideways or back away (here, look direction is matched to movement direction). With sequential training, the designer first demonstrates an example input path to represent the main character. Then, the system replays the main-character's motion path while the designer demonstrates the sample reaction. For behavior generation, the user simply controls the main character and reaction is generated in real time.

3.2. Tabletop Tangible User Interfaces (TUIs)

Demonstrating interactive behaviors is reminiscent of acting or puppetry; actions generally done in the physical world, away from traditional computers. Using a mouse mapped to an on-screen avatar removes this direct physicality and forces an artist to map their intentions through the arguably less-than-intuitive mouse interface, separating the artist action and perception space [FBR86, SWK*04].

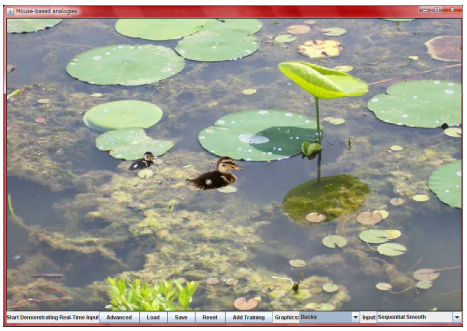


Figure 2: *The mouse GUI. Notice the large work area and lack of parameters and settings.*

Figure 3: *A user interacting with the Vicon TUI system.*

Our Tangible User Interfaces (TUIs)-based tabletop design (see Figure 3) allows artists to use physical objects (the TUIs) as handles to demonstrate behavior. The TUIs offer immediate physical feedback of the system state, providing intuitive interaction not possible with traditional interfaces [IU97]. Using TUIs also allows for character orientation, and multiple TUIs can be used for simultaneous training of both the main and reacting character behaviors, either two-handed by one designer or by a pair of collaborators.

4. Algorithm

During training, the system simply stores the paired motion data. During generation, the algorithm compares the runtime situation between the main and reacting character to the training data. The training most similar to the current situation is used to direct the generation of the next character output; the overall generation is based on a mix of training data from various source locations. This method reacts immediately to changing user input and avoids large chunks of consecutive, static replay of training data while maintaining the style and properties of the demonstration.

The key problems to be solved are to a) select a similarity metric that matches behavioral similarity as end-users may recognize it, and b) ensure the generation algorithm maintains the characteristics and textures given in the training data. All of this must happen in real time.

4.1. Algorithm Formalization

The variables I, R represent time-series data for the main (I) and reacting (R) characters. Following, I^t, R^t are the training time-series data used by the generation system, recorded from the demonstration. Pseudo code is given below: `BestMatch()` finds training data most similar to the current situation, and `Generate()` uses the selected data to direct the next output action. New user input is not used until the subsequent iteration because features used (e.g., relative character position) require data for both characters. Our implementation operates at 40Hz so this delay is negligible.

```
Loop
  e = BestMatch(I^t, R^t, I, R)
  newMovement = Generate(e)
  R.append(newMovement)
  I.append(getNewInput())
```

4.2. Data and Features

Our data is a time-dependent array containing the location, x,y, and the direction d of each entity at each time point. We extract various features that decide what characteristics

of behavior will be matched. We generally ignore world-coordinates and focus on the relationship between the two entities and changes in local state, and these features are evaluated over a time window to encapsulate a trend over time. We explored many features not discussed here (e.g., direct path data, distance and delta distance between characters), and settled on the following features, as illustrated in Figure 4. We omit a detailed discussion on the omissions for brevity. The use of these features is different for each algorithm step as outlined in each respective section.

Velocity – the magnitude of the vector between an entity's position and its previous one. This captures speed and acceleration-related aspects of behavior such as different reactions for stopped, accelerating, or slow input.

Relative position – position of the reactor in relation to the main character's position and look direction (coordinate space). This captures relational behavior such as following, circling, and approaching. One scalar value per axis: representing how much the reacting character is behind or in front of, and to the left or right of the main character.

Normalized look direction – look direction normalized to movement direction, with 0 pointing forward. This is the relationship between where a character is looking and moving, e.g., if it is backing up or moving sideways.

Relative look direction – the difference between the entities' look directions. This captures turning away shyly when observed or aggressively facing an opponent.

Absolute Movement direction – the vector from the previous world-coordinate position to the current one. This does not involve the other entity but is used in generation (Section 4.4) to add high frequency texture to the output.

ΔDirection – change in direction from one step to the next, represents the shape of the locomotion path (not in relation to the other entity). This feature helps to identify similar movement shapes and styles such as shaky or smooth.

4.3. BestMatch (Similarity Metric)

Our similarity metric is heavily based on Image Analogies [HJO*01, HOCS02] but applied to dynamically generated motion sequences. This metric has two key components: overall situational similarity and generated path coherency. These are checked in parallel and combined in each step over a given movement-history neighborhood n (40 samples (1s) in our implementation).

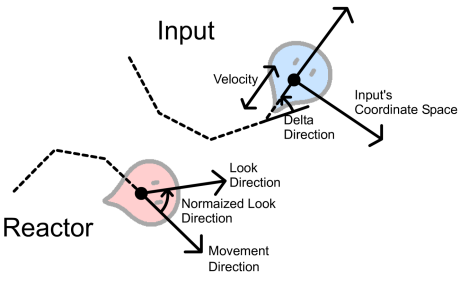
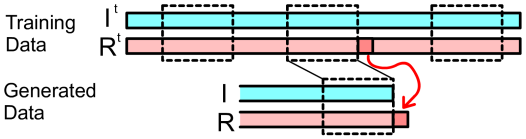


Figure 4: *Data features: All features except relative position are on both entities, but only shown on one for image clarity.*

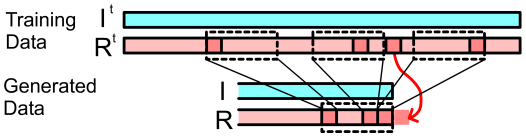
Situation Similarity – This is based on the relationship between the two entities, using the relative position (input-character centric), relative look direction, and velocity features. It compares the n most recent pieces of user input and generated output from I,R to a moving window of size n over the training data I^t,R^t (Figure 5(a)). At each window location the features from each of the four paths form multi-dimensional vectors and corresponding vectors (I vs I^t, R vs R^t) are compared using Euclidean distance squared. These distances are then summed over the window, providing a measure of similarity at that window location. A smaller value represents a better match.

Generated Path Coherency – This emphasizes the shape, style and features of the generated path R in relationship to the trained R^t while putting less emphasis on the relationship between entities I and R. This relationship is still important, however, as some aspects of coherency depend on the relationship between the entities, such as when the reacting entity wants to finish a circle around the main entity it must properly follow as the main character moves. This metric uses normalized look direction, delta direction, relative position, and velocity. When there is no training data that matches well to the current inter-entity situation (i.e., situation similarity is weak) generated path coherency helps to ensure a generation that matches the characteristics of R^t.

This metric compares the recently generated data from R over n to the regions in R^t that were used to generate the recent R (the entire R^t is not searched). That is, given recently generated elements R_k over the n, the source neighborhood in R^t that was originally used to generate R_k is compared to the most recently generated R (Figure 5(b)). The intuition



(a) Situation Similarity compares recent real-time data to the entire dataset.



(b) Generated Path Coherency examines the regions of the training data recently used in generation.

Figure 5: *Metrics derived from Image Analogies [HJO*01]*

here is to continue a patch from a previous similarity match if the current similarity match is weak.

Similarity Balancing – The Image Analogies algorithm [HJO*01] combines the two similarity metrics by statically weighting them with a coefficient k to add bias; the metric with the best weighted score is selected for that step. This did not work with our application and resulted in a problem we call coherence loops: when coherence match is used to generate output for several consecutive steps then the result of generation, by design, will be increasingly similar to the training data. The improving coherence match is eventually exclusively used, with situation similarity being ignored, and the reacting entity starts to loop through sections of R^t. This issue does not occur in Image Analogies and Curve Analogies because all data is given at the beginning, allowing the use of multi-resolution approaches. Multi-resolution is difficult in our system, however, as we are generating in real time and cannot look ahead in our input data. We initially tried to mesh both metrics in several ways (e.g., average, trend-vs-detail, staggered sample rate), but did not get satisfactory results. Our solution is given below, but this is a rich area for future work.

We change the previously-static weight coefficient k to a dynamic value to target a situation-similarity-to-coherency match ratio t. Then, k is automatically tuned each generation step to bias the results so that over time we keep t balanced. In our implementation, we use a 1:1 target ratio and k is tuned linearly. A similar algorithm is used in texture synthesis systems to match the overall color histogram [KFCO*07]. Following, the data from R^t immediately following the source region is passed to the generation system. One problem with this balancing approach is noise. Instability in the similarity metrics (jumping between regions) and switching rapidly between situation similarity and coherence can cause large, rapid variations in the source data passed to the generation system, resulting in distracting rapid character movements. We explain our solution below.

4.4. Output Generation

The generation system receives the piece of training data (target data) to be used from `BestMatch` and generates the next entity output. The naïve approach is to simply copy this data directly to the output as in texture synthesis. The problem with this is that many features depend on history and are relative to the other entity and it is impossible to solve a movement that matches all features. Also, when the training data jumps between drastically different states over consecutive steps this approach does not provide a meshing mechanism to generate intermediate data. These jumps suggest that transitions are missing from the training data, and the generator function must handle discontinuities while maintaining the texture, personality, and character that was demonstrated.

Our generation approach, a key technical contribution of this paper, is to decompose emotion (the motion) into its low-frequency (intentional move to certain relational position) part and high-frequency (texture of the motion) part and treat them separately. While Fourier analysis has been done on motion paths before (e. g., [UAT95]), we are the first to use it in terms of emotion and personality.

General Trajectory Generation – Motion is generated using relative position, normalized look direction, and velocity. Normalized look direction is copied directly to output, and a vector is constructed to move the entity from its current location to the target relative position and then scaled to the target velocity (Figure 6). Although the entity moves toward the target position rather than be at that position, the velocity scaling in combination with the high generation rate helps to create very convincing results. Here we deal with the noise resulting from the `BestMatch` instability by applying a simple linear smooth (average) over a history of three samples. The results of this are very convincing and result in a more stable, consistent generation. The problem, however, is that by removing the high-frequency noise we also remove the high-frequency data, such as the movement detail and texture. We implemented a fix for this described in the next section.

Another problem is that, even with smoothing, normalized look direction is very noisy. This happens because of the nature of the normalized look direction itself: if a character keeps a static look direction in the world, but rapidly changes movement direction, the normalized look direction (based on movement direction) changes rapidly between drastically different values. An entity moving rapidly forward and then backward has data that alternates between 0 and π. Our solution to this is to limit rate of change of the actual world-coordinate look direction. This lowers the amount of noise in resulting look direction, but some jitter remains. This is an important problem for future work.

Detail Incorporation – To restore detail removed by smoothing we do frequency analysis using Haar wavelets, extracting the high-frequency detail from the target and directly incorporating it into the output. We apply Haar decomposition on the motion direction feature as this captures path texture irrespective of velocity. A single application of the discrete Haar decomposition scales our path data to half resolution and stores the removed high-frequency detail separately. This gives a frequency cut at $f_s/2$ where f_s is the

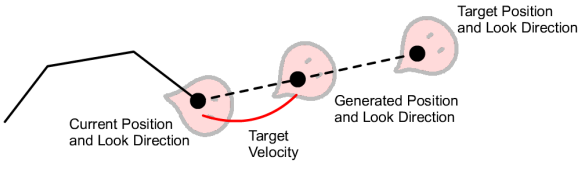


Figure 6: *To avoid drastic jumps, the reactor moves towards the target position with the velocity taken from training data.*

sampling rate: given k cumulative decompositions, this cut is at $f_s/2^k$. The resulting k high-frequency datasets (one per decomposition) can be re-composed to form a single high-frequency-only signal that we use in our generation. Our system uses four-level Haar decompositions, a frequency cut of $f_s/16$, or about 2.5 samples /s We found this to capture sufficient detail without affecting general trajectory.

The high-frequency data from the target is used to perturb the generated (but smoothed) trajectory. While smoothing compensated for `BestMatch` instability in trajectory generation, high-frequency source data cannot be smoothed in the same fashion and so this instability remains, where interweaving detail from rapidly alternating training locations results in noisy output not coherent to the training. Our solution applies the detail in patches of 16 samples: a patch is used in subsequent steps until the end when a new patch is selected. These are 0.4 seconds long so the delay between changed behavior and matching path detail is minimal.

5. Evaluating Puppet Master

Our evaluation consisted of two parts. The first part, the *artist* study, asked participants to design new behaviors using our system. The second part of the study, the *end-user* study, asked participants to interact with behaviors created in the first study. Our goals were to identify weaknesses in our algorithm and interface, to get general user feedback and to determine how much (and what sorts) of characteristics, emotions, and personality traits are captured by our system.

We initially conducted a pilot study to evaluate the study protocol and procedure. 5 participants (2 female, 3 male) joined the *artist* pilot and 2 participants (1 male, 1 female) joined the *end-user* pilot. These pilots exposed language and questionnaire wording that was confusing or strongly biased users toward particular responses.

5.1. Experimental Testbed

The experiments used the tabletop TUI interface (Section 3.2) running a Pentium 4, 3.0 GHz PC, maintaining 40fps for up to 80s of training data (a behavior generally requires less, as described below). We used the graphics in Figure 7 as static textures and a SMART Technologies 4'10" x 3'7" high-resolution (2800x2100) rear-projected tabletop display with pucks on the top to control input (Figure 3). The pucks are tracked at 100fps by a six-camera Vicon motion-tracking system, and the character graphics are drawn on the tabletop surface directly below the pucks in real time.

5.2. Methodology

Participants – Twenty students (10 per study) from varying disciplines were selected from our university population and paid $15 for participation. All users reported some to extensive programming experience and strong confidence with computers. In the *artist* study (2 female, 8 male), four participants reported artistic experience with three having formal training and one identifying herself as an artist, and three users reported basic animation experience. Ages ranged from 19 to 32 (M=22.8, SD=3.8). In the end-user study (4 female, 6 male), nine participants reported artistic experience with five identifying themselves as artists, and four users reported animation experience (two extensive). Ages ranged from 19 to 27 (M=23.7, SD=2.71). All participants had no prior exposure to the system and no participants from the *artist* study took part in the *end-user* study.

Procedure – The artist study explored how general users can use our system to create interactive behaviors. In one-hour sessions, participants were first asked to design five particular interactive character behaviors given the following keywords: *lover, bully, playful friend, stalker,* and *afraid*. Participants completed a short written survey about the result and experience after each behavior. Following, we evaluated the internal validity of the design by loading the five created behaviors each participant created in a scrambled order (fixed across participants) and asking them to interact with, and recognize, each behavior. Participants were not notified ahead of time that they would be revisiting their own designed behaviors.

The *end-user* study observed how general users react to the behaviors created using our system, and whether a sense of character emotion and personality emerged. We subjectively selected five behaviors created by participants in the *artist* study (one per each of the five behavior types), and participants were asked to "interact with and explore the characters" for each behavior in a fixed order, and to "describe the character" in a questionnaire. Care was given to avoiding affective or anthropomorphic language when presenting the task to the end users, avoiding words such as "personality", "behavior", and "emotion". In the second part participants were asked to interact with a set of "other" behaviors which were in-fact a scrambled set of the previous behaviors. This time users were asked to match each of the behaviors to the list of "correct" behaviors as given in the *artist* study.

5.3. Artist-Study Results

Eight of 10 users in this study identified 100% of their own behaviors. Further, in 74% of the cases (using 5 point Likert) users agreed or somewhat agreed they were satisfied with the resulting behavior, and in 22% of the cases they neither

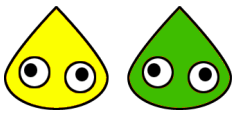


Figure 7: *The graphics used in our evaluation, designed generically to avoid particular emotion or personality.*

agreed nor disagreed. The mean training time of accepted behaviors was 32.5s (SD=18.0s, min=9s, max=85s). The average number of trials required before accepting a behavior was 1.7 (SD=0.9, mode=1 at freq.=56%, max=4 trials). The average amount of time a user spent testing a result before accepting it was 70.0s (SD=68.2s). In 46% of the cases users disagreed that the generated behavior felt mechanical with 26% neither agreeing nor disagreeing. In 48% of the cases users agreed that the behavior felt human-controlled (42% somewhat) with 26% neither agreeing nor disagreeing.

In the post-test questionnaire, on seven-point Likert, all 10 *artist* users agreed (5 strongly) that they enjoyed using the system, while 7 disagreed that the system was frustrating to use (1 strongly, 2 somewhat), all reported that the characters were fun to play with (6 strongly, 2 somewhat) and 6 users reported that movement jitter was distracting. The two users who failed to recognize their own designed behaviors were also the only two users who did not use puck orientation during behavior training, resulting in poor quality behaviors.

Four artist users were notably immersed in the interface. Some made exaggerated faces, noises, and spoke to the characters while training. One artist used the "jaws" theme while training the "afraid" behavior, and another commented "what a jerk!" when observing their "bully" character. Users generally expressed excitement about and satisfaction with the capabilities of the system: "the system responded accurately and behavior was smooth, human-like, with a human touch", "it's even a better stalker than I am!", "it almost looks as if someone is controlling it.", "it did exactly as I wanted! Very entertaining! (maybe it's just me?)", "nailed it!", "I like it! I can see its bright future", "the playful friend is a hoot!" Several users commented on the robustness of the system, and one user was excited that it "even reacted consistently with what [he] thought of after the fact." Also, users enjoyed the tabletop system, finding it "super easy and intuitive to operate. Instant results."

On the other hand, several users reported issues with the system, commenting on the resulting generation as well as the simplicity of our system: "it felt a bit mechanical with some movements", "as complexity of behavior rises it feels more mechanical", "if you pause to catch your breath, the system takes it as deliberate behavior", "I need to try more complicated behaviors", "this setup cannot interpret smaller actions that well", "he doesn't have hands so I can't punch", "difficult to imagine what one pretty slime does to bully another pretty slime." Further, six of the ten users had issues with occluding the Vicon markers on the controller puck.

5.4. End-User-Study Results

In the first part of the end-user study users were simply asked to interact and describe prototype characters (without being prodded to look at behaviors or emotions). Here, on a six point scale titled "the character felt..." ranging from "extremely mechanical" (1) to "somewhat mechanical" (3, 4) to "not mechanical at all" (6) the average response across all behaviors was 4.04 (SD=1.19,Mode=4 at 36% frequency). On another scale ranging from "a human is controlling it" (1) to "somewhat lifelike" (3, 4) to "not lifelike at all" (6), the average response was 3.4 with a mode of 5 at 24%.

To our pleasant surprise, out of the 50 cases (5 behaviors across 10 participants), characters were identified using the exact keywords used in the artist study 9 times, and another 10 times using very similar words (for example, "girlfriend" instead of "lover", "naughty, trying to bug me" instead of "bully"). Out of the 10 participants, 2 did not match any behaviors, 2 matched 1 behavior, 3 matched 2 behaviors, 1 matched 3 behaviors, and 2 users matched 4 behaviors correctly. Furthermore, in the open-ended questionnaires 52% of all end-users behavior descriptions were using social and behavioral descriptions (28% purely social), 34% of all the descriptions were using mechanical language (18% purely mechanical), with 14% being roughly a half-half mix. For the second part of the *end-user* study, participants were asked to match the five behaviors against the original keywords used. The results are given in Table 1, with the diagonal showing the number of end-users, out of 10, who matched the pre-designed behavior to its exact keyword.

On the final questionnaires, 4 users agreed that the characters were sometimes confusing (1 somewhat), 1 neither agreed nor disagreed, and 5 users disagreed (1 strongly, 1 somewhat). One strong observation throughout the study was that users tended to see social characteristics and used anthropomorphic language. For example, end users mentioned: "the guy who kept sucker-punching", "each one could bring to mind some real-life analogy", "he needs more confidence", "I liked the part when it came close to my character ... kind of like a dog who is happy to see you", "He keeps trying to either hit you or kiss you", "like an annoying kid brother in my face", "he [the stalker] seemed like he wanted to approach me, but he was too shy", "facing it and watching it panic like it had been discovered somewhere where it shouldn't be was fun", "she [playful friend] is like a little sister who wants to talk to me."

Users were asked on the final questionnaire to describe the things they liked and disliked about each character. While

Matched to \ Actual Trained Behavior	Lover	Bully	Playful Friend	Stalker	Afraid
Lover	6	1	3	0	0
Bully	0	5	4	0	1
Playful Friend	4	3	2	1	0
Stalker	0	0	1	6	3
Afraid	0	1	0	3	6

Table 1: *How behaviors were matched to original designs.*

some of these comments were analysis oriented, such as "actions were vague, subject to interpretation", many of the comments referred to the participant's opinion of the character's personality. For example, for the afraid character (which stayed away from the participant's character) one user wrote "I didn't really like anything, didn't even give me a chance to get to know him", and others complained that it "tries to invade my personal space. I like a nice personal space bubble", or "it doesn't feel friendly!"

Similar to the artist study, some participants commented that the characters felt a bit fake when the jitter was too noticeable and several participants complained that the personalities were too simple: "the personalities were very blunt, they were easy to see", "I wish they could touch each other". All end-user participants enjoyed the experiment (6 strongly agreeing). 7/10 users reported the pucks frustrating to use (all of these users commented on how easy it was to occlude the Vicon markers), with the remaining 30% disagreeing or strongly disagreeing. However, several users commented that the table was "easy to use" and "intuitive".

6. Discussion

These results help to support our claims about Puppet Master. The fact that 80% of the artist participants recognized 100% of their behaviors and were satisfied with the results suggests that our algorithm successfully supports some level of expression, and captures sufficient personality-related characteristics for recognition by the designer. That this was accomplished without training at on average 32.5s shows, even for outliers (e.g., the 85s case), our algorithm allows artists to create recognizable behaviors in significantly less time than it would take to program. Finally, that this was accomplished in on-average 1.7 attempts by novices shows that designers are able to satisfactorily and easily create behaviors, and that the real-time re-training and generation enabled the artists to flexibly explore design possibilities.

The end-user part of our study demonstrated that in 38% of the cases behaviors not only emerged but closely matched the artist keywords, based on motion only (Table 1). We believe that this supports our claim that our algorithm captures the personality and style of the demonstrated behavior. Further, the results in Table 1 seem to hint at crosstalk between similar behaviors: for example, afraid and stalker are often mistook for each other while lover, bully, and friend are rarely mistaken for stalker or afraid. This shows that, even in the cases where behaviors are not matched properly, there is still a strong component of feeling and style captured from the demonstrated data.

Both studies suggest a strong sense of user engagement. The explicitly positive study results, the verbal excitement, as well as the extensive use of social and anthropomorphic language suggests that the participants were interested and mentally involved with the design process.

7. Limitations and Future Work

Our current implementation does not handle dynamics over a large time scale and will fail to accurately represent an angry character gradually calming down. Perhaps this could be explored through high-level features and multi-resolution searches. It is also useful to consider how this work can combine with other behavior models and systems for a multi-level solution, and to understand the absolute limitation of our approach, that is, when high-level behavior changes may be better designed using scripting and explicit states.

Accounting for environmental issues (wall, tree, etc) would improve the versatility of Puppet Master. Related, extending to several simultaneous entities would allow swarm-like behavior with individual personalities, e.g., to train a group of archers and knights to storm a castle. Systems that learns crowd behavior from examples [LCHL07, LCL07] mainly focuses on collision avoidance while we want to allow artists to interactively design more intentional crowds.

Perhaps our work could be applied to the design of robot behaviors, e.g, replacing physical pucks and tabletop displays with mobile robots such as curlybot [FSMI00]. Physical movement can give stronger impression of personality and emotion than virtual characters, but introduces severe constraints on motion (e.g. robot cannot jump to distant position). However, many of the techniques developed in this work such as similarity-coherence balancing and trajectory-texture separation should be applicable to real robots, too. Our ultimate goal is to design all aspects of character behavior, not just locomotion, responding to various input such as dancing to music, sword-swing against an opponent, and meow noise of a cat responding other meow noise.

8. Conclusion

Believable, convincing, and stylistic interactive behavior is an important aspect of any computerized entity that must interact with humans, such as avatars, video game characters, or even robots. Traditionally, the creation of such a system has been left to logical, step-by-step computer algorithms; tools generally out-of-reach for non-technical artists and ill-designed for the creation of stylistic behaviors. In this paper we presented the first system that enables the programming of interactive behaviors by demonstration with real-time generation, making the creation of believable, stylistic interactive characters accessible to the non-technical artist.

Acknowledgements

This work supported in part by NSERC, iCore, JSPS, JST, and the University of Calgary.

References

[ABC] AMAYA K., BRUDERLIN A., CALVERT T.: Emotion from motion. In *Proc GI '96*, pp. 222–229.

[Bat94] BATES J.: The role of emotion in believable agents. *Comm. ACM 37*, 7 (July 1994), 122–125.

[BG95] BLUMBERG B. M., GALYEAN T. A.: Multi-level direction of autonomous creatures for real-time virtual environments. In *Proc. SIGGRAPH '95* (1995), pp. 47–54.

[Bre02] BREAZEAL C.: *Designing Sociable Robots*. MIT Press, 2002.

[Cyp] CYPHER A.: Eager: programming repetitive tasks by example. In *Proc. CHI '91* (NY), pp. 33–39.

[DE05] DINERSTEIN J., EGBERT P. K.: Fast multi-level adaptation for interactive autonomous characters. *ACM Transactions on Graphics 24*, 2 (2005), 262–288.

[DEV] DINERSTEIN J., EGBERT P. K., VENTURA D.: Learning policies for embodied virtual agents through demonstration. In *Proc. IJCAI '07*, pp. 1257–1262.

[DYP03] DONTCHEVA M., YNGVE G., POPOVIĆ Z.: Layered acting for character animation. *ACM Trans. Graph. 22*, 3 (2003), 409–416.

[FBR86] FJELD M., BICHSEL M., RAUTERBERG M.: Build-it: a brick-based tool for direct interaction, 1986.

[FSMI00] FREI P., SU V., MIKHAK B., ISHII H.: curlybot: designing a new class of computational toys. In *Proc. CHI '00* (NY, USA, 2000), ACM, pp. 129–136.

[HGP04] HSU E., GENTRY S., POPOVIĆ J.: Example-based control of human motion. In *SCA '04* (Switzerland, 2004), EG, pp. 69–77.

[HJO*01] HERTZMANN A., JACOBS C. E., OLIVER N., CURLESS B., SALESIN D. H.: Image analogies. In *Proc. SIGGRAPH '01* (2001), pp. 327–340.

[HOCS02] HERTZMANN A., OLIVER N., CURLESS B., SEITZ S. M.: Curve analogies. In *Proc. EGRW '02* (2002), pp. 233–246.

[IMH05a] IGARASHI T., MOSCOVICH T., HUGHES J.: Spatial keyframing for performance-driven animation. In *Proc. SIGGRAPH '05* (July 2005), pp. 107–116.

[IMH05b] IGARASHI T., MOSCOVICH T., HUGHES J. F.: As-rigid-as-possible shape manipulation. *ACM Trans. Graph. 24*, 3 (2005), 1134–1141.

[IU97] ISHII H., ULLMER B.: Tangible bits: towards seamless interfaces between people, bits and atoms. In *Proc. CHI '97* (NY, 1997), ACM, pp. 234–241.

[Kas82] KASSIN K.: Heider and simmel revisited: causal attribution and the animated film technique. *Rev. Pers. Soc. Pschol. 3* (1982), 145–169.

[KFCO*07] KOPF J., FU C.-W., COHEN-OR D., DEUSSEN O., LISCHINSKI D., WONG T.-T.: Solid texture synthesis from 2d exemplars. *ACM Tran. Graph. 26*, 3 (2007), 2:1–2:9.

[LCHL07] LEE K. H., CHOI M. G., HONG Q., LEE J.: Group behavior from video: a data-driven approach to crowd simulation. In *SCA '07* (Switzerland, 2007), EG, pp. 109–118.

[LCL07] LERNER A., CHRYSANTHOU Y., LISCHINSKI D.: Crowds by example. *Comp. Graphics Forum 26*, 3 (2007), 655–664.

[LL04] LEE J., LEE K. H.: Precomputing avatar behavior from human motion data. In *Proc. SCA '04* (Aire-la-Ville, Switzerland, Switzerland, 2004), Eurographics Association, pp. 79–87.

[Mae95] MAES P.: Artificial life meets entertainment: lifelike autonomous agents. *Comm. ACM 38*, 11 (1995), 108–114.

[MMMI05] MATSUI D., MINATO T., MACDORMAN K. F., ISHIGURO H.: Generating Natural Motion in an Android by Mapping Human Motion. In *Proc. IROS '05* (USA, 2005), IEEE, pp. 1089–1096.

[MWK89] MAULSBY D. L., WITTEN I. H., KITTLITZ K. A.: Metamouse: specifying graphical procedures by example. In *Proc. SIGGRAPH '89* (NY, USA, 1989), ACM, pp. 127–136.

[Nor04] NORMAN D. A.: *Emotional Design*. Basic Books, NY, 2004.

[PG96] PERLIN K., GOLDBERG A.: Improv: a system for scripting interactive actors in virtual worlds. In *SIGGRAPH '96* (NY, 1996), ACM, pp. 205–216.

[Rey87] REYNOLDS C. W.: Flocks, herds and schools: A distributed behavioral model. In *Proc. SIGGRAPH '87* (NY, 1987), ACM, pp. 25–34.

[RN96] REEVES B., NASS C.: *The Media Equation*. CSLI Publ., UK, 1996.

[RPI04] RAFFLE H. S., PARKES A. J., ISHII H.: Topobo: a constructive assembly system with kinetic memory. In *Proc. CHI '04* (NY, USA, 2004), ACM, pp. 647–654.

[SWK*04] SHARLIN E., WATSON B., KITAMURA Y., KISHINO F., ITOH Y.: On tangible user interfaces, humans and spatiality. *Personal Ubiquitous Comput. 8*, 5 (2004), 338–346.

[TBvdP04] THORNE M., BURKE D., VAN DE PANNE M.: Motion doodles: an interface for sketching character motion. In *Proc. SIGGRAPH '04* (New York, NY, USA, 2004), ACM, pp. 424–431.

[UAT95] UNUMA M., ANJYO K., TAKEUCHI R.: Fourier principles for emotion-based human figure animation. In *SIGGRAPH '95* (1995), pp. 91–96.

[WH97] WILEY D. J., HAHN J. K.: Interpolation synthesis of articulated figure motion. *IEEE Comp. Graph. and App. 17*, 6 (/1997), 39–45.

[Wol97] WOLBER D.: Pavlov: an interface builder for designing animated interfaces. *ACM TOCHI 4*, 4 (1997), 347–386.