Style by Demonstration for Interactive Robot Motion

Jeffrey Allen,[1] James E. Young,[2] Daisuke Sakamoto,[1,3] Takeo Igarashi[1,3]
jeffrey.c.allen@gmail.com, young@cs.umanitoba.ca, sakamoto@designinterface.jp, takeo@acm.org
[1]JST ERATO Tokyo, Japan
[2]University of Manitoba Winnipeg, MB, Canada
[3]The University of Tokyo Tokyo, Japan

ABSTRACT
As robots continue to enter people's everyday spaces, we argue that it will be increasingly important to consider the robots' movement style as an integral component of their interaction design. That is, aspects of the robot's movement which are not directly related to a task at hand (e.g., pick up a ball) can have a strong impact on how people perceive that action (e.g., aggressively or hesitantly). We call these elements the movement *style.* We believe that perceptions of this kind of style will be highly dependent on the culture, group, or individual, and so people will need to have the ability to customize their robot. Therefore, in this work we use *Style by Demonstration*, a *style* focus on the more-traditional programming by demonstration technique, and present the Puppet Dancer system, an interface for constructing paired and interactive robotic dances. In this paper we detail the Puppet Dancer interface and interaction design, explain our new algorithms for teaching dance by demonstration, and present the results from a formal qualitative study.

Author Keywords
Human-Robot Interaction, Human-Computer Interaction, Style by Demonstration, Qualitative Evaluation.

ACM Classification Keywords
H.5.2 [**Information systems**]: Information Interfaces and Presentation – *User Interfaces.*

INTRODUCTION
As the field of robotics continues to advance, more robots are being developed for public and domestic spaces: robots already help with housework (e.g., the iRobot Roomba robotic vacuum), provide patient companionship [1] and perform menial tasks [2] in hospitals. In these personal contexts, given that people tend to anthropomorphize robots they encounter in everyday life [4, 23, 27], we believe that people will be particularly sensitive to the robot's movement style when completing given tasks: we use style to mean a robot's "expressive movement…the way in which behavior is performed" [10]. For example, a robot that quickly lunges at a person to perform a handshake would likely be perceived as being hostile, while one which moves slowly may be seen as polite.

Figure 1. Dance partners: the leading puppet (cat, left) and following robot (dog, right). The user puppeteers the leader, and the follower interactively dances with the leader.

Although this example may be extreme, it illustrates an important point of robot behaviors that interact with people: non-critical style movement variables outside the primary task or movement goal, such as movement speed, texture, or overall path, will impact how people perceive the interaction. We argue that this task-superfluous style layer of the robot's movement behavior will be important to help robots appropriately "fit into" the social roles that people give them [15, 24].

We believe that people's preferences for a robot's interaction style are likely to vary widely, and be very sensitive to the individual's culture and tastes. As such, a challenge will be to tailor style to the individual; this is reminiscent of the popular customizations available for other personal electronics, such as skins, covers, or themes. We approach this problem by using programming by demonstration, a popular robot behavior authoring method where people are able to create robot behaviors simply by performing an exemplar, similar to how they may teach a person [6, 12]. Unfortunately, most programming by demonstration work either focuses on the goal only, ignoring the style, or does not produce an interactive result, severely limiting which robotic behaviors it can be used for. New algorithms and techniques need to be developed for the creation of the style component of *interactive* robot behaviors. We call this approach Style by Demonstration (SBD) simply as a means to differentiate it from the

DIS 2012, June 11-15, 2012, Newcastle, UK.

broader programing by demonstration, to explicitly target style.

In this paper we present exploratory work with the primary purpose of building an SBD system and investigating how people interact with and engage this class of interface. We take an informal approach to defining style in this work, where we note that the notion of robotic movement style as we use it is intuitively understandable but perhaps more difficult to rigorously define. While we note that this challenge will need to be addressed to properly explore the SBD domain in depth, our less formal approach to defining *style* is sufficient to serve our current exploratory and initial-prototype purposes outlined below.

Ultimately, SBD will only be one layer of the behavior-creation problem, and will have to integrate seamlessly with robot sensor data and hard environmental and situational constraints that restrict movement. For this project, we target interactive robotic dance as an initial step toward the larger integrated vision as: a) dance is heavily style oriented and has fewer higher-level task aspects to be considered, b) with our setup there are very few task-oriented constraints (e.g., movement limitations, obstacles) to integrate, and c) paired dance is highly interactive with real-time input, just as we envision many robotic tasks will be.

We present *Puppet Dancer* (Figure 1), an interactive robotic platform for authoring interactive, paired robotic dances by providing a demonstration. A user can teach one robot (the *follower* dog) how to interactively dance in real time with a puppet (the *leader* cat) by providing a demonstration of the desired dance (**Figure 2**, left side). Next, when the puppet *leader* is manipulated, the *follower* will automatically dance in the fashion demonstrated (**Figure 2**, right side). We further present a formal qualitative study which explores the overall question of how users will engage and hopefully use our SBD interfaces, and tests various components of the interface designs and the effectiveness of the underlying algorithm.

This paper's contributions are: 1) an SBD interface design and implementation for authoring interactive robotic dances, 2) extensions of prior SBD work to achieve quality SBD results, and 3) a formal qualitative study on how users engage SBD.

RELATED WORK

There have been several related projects which explore the importance of a robot's interactive movement style. It has been shown how the movement style of a search and rescue robot can impact the emotional state of victims [5], how the movement style of a vacuum robot can portray personalities [21, 26], and how even the movement of an abstract physical object such as a stick is enough for people to construct elaborate personalities [13]. Puppet Dancer builds on these results to create robots which portray interactive dance styles through their reactive movement patterns.

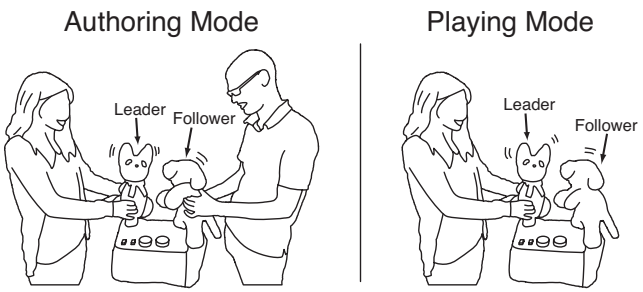

Figure 2. Modes of interaction: *authoring mode* (left) where demonstrations are performed, and *playing mode* (right) where the *follower* dance is automatically generated.

Programming by demonstration is common for generating robot behaviors, e.g., for teaching navigation routes [14] or performing specific physical tasks [11, 12, 16, 19], and generally does not handle the *style* aspect of behavior. Notable exceptions include robots which learn human-like motions and poses [17] or explicitly learn stylistic movements [9, 20], although these copy and reproduce demonstrated movements verbatim and do not learn the interactive aspects of the behavior as with our dance. Two prior interactive SBD projects exist [25, 26] which learn interactive locomotion paths: we extend this work to the entirely different feature set of robot dancing.

Robot customization has also been explored for the static physical appearance. The AIBO robot is packaged with stickers for customization and decoration, and people have been found to decorate (and even buy clothes) for their iRobot Roomba robotic vacuum cleaner [22, 23]. Thus we believe that the kinds of customization enabled by SBD and our Puppet Dancer system will be of interest to end users.

Entertainment robots, such as the Puppet Dancer platform, is an emerging market and research area that has received a great deal of attention, for example, with products such as the Sony AIBO, the WowWee line (e.g., the Robosapien), and Keepon the dancing robot [18]. While some of these, such as the AIBO, do support rudimentary learning, we are not aware of any platform which can learn the style of an interactive behavior from a demonstration.

PUPPET DANCER

Puppet Dancer is a proof of concept for SBD, to demonstrate how people can teach interactive movement style to robots. The base idea is interactive, paired dance: there is a *leader* cat puppet and a *follower* dog robot (Figure 1, 2), where the *follower* moves in direct real-time response to the *leader's* dance. For example, if the *leader* does a dip the *follower* may dance from side to side.

A user can teach their desired interactive dance to the *follower* robot by simply playing with the stuffed toys to perform a paired dance, manipulating both the *leader* puppet and *follower* robot simultaneously (**Figure 2**, left). The *follower* monitors the exemplar dance and learns the interactive movements. Then, the user can puppeteer the *leader*, and the *follower* automatically dances interactively

(**Figure 2**, right). Thus Puppet Dancer has two explicit modes: the *authoring* mode, where the person provides a demonstration of their desired interactive dance for the robot to learn, and the *playing* mode, where the person freely performs a dance using the *leader* puppet and the *follower* robot interacts using the learnt behavior. We expect that the user's movements during *playing* will not be identical to those given during *authoring*, so Puppet Dancer must be robust enough to adapt in real-time to the user's given dance input.

The user can switch seamlessly between the *authoring* and the *playing* modes to provide additional training to the *follower*, for example, to add new interactive dance moves or to add additional examples of existing moves. The mode change is enabled via a toggle button (**Figure 3**) with the label "training" on it. When pressed, the training button starts blinking and stays depressed, and the system enters *authoring* mode. Pressing the button again makes the light extinguish and the button pops out, and the system is in *playing* mode. In addition, there is a "reset" button which, when pressed, makes the *follower* forget all training learned.

Users can give real-time feedback during the *playing* mode to modify how the *follower* uses training data to improve the final result. This reinforcement-learning interface mechanism is enabled through two arcade-style press buttons mounted on the top of the Puppet Dancer platform (Figure 1, **Figure 3**). The user can press the green *approve* button to signal to the *follower* when they like the dancing, and can press the red *disapprove* button to signal dislike.

Inspired by Human-Human Interaction

The Puppet Master interface design was heavily motivated by how people interact when teaching each other. While a learner is trying a new task such as playing a guitar chord, after initial instruction the teacher will give regular verbal feedback as guidance, e.g., saying "yes" or "no" as the learner tries to shape the chord. We modeled this as our reinforcement-learning like/dislike buttons. In addition, we noticed how the teacher will sometimes stop the learner and provide additional demonstration, e.g., to teach the learner how to strum with one hand while holding the chord on the other, or to refine existing skills, e.g., saying "like this" while giving a physical demonstration. We modeled this aspect as the ability to add additional training at any time by entering the *authoring* mode.

IMPLEMENTATION AND ALGORITHM

Both the dog *follower* and the cat *leader* in our dance interface (**Figure 1**) are constructed from Robotis Bioloid kits. As shown in **Figure 4**, there is a 4 DOF spine made of servos inside the stuffed animals. For both, this spine can sense the shape of the dancer to record motions during *authoring*. For the dog *follower*, this spine provides the actuation required to perform the generated dance; the actuators have been disabled for the *cat* leader to make the spine easier to manually manipulate.

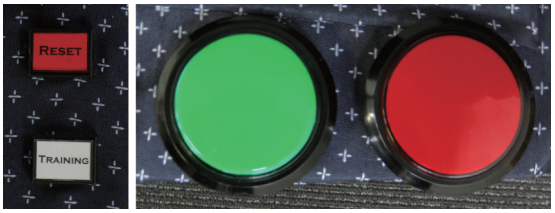

Figure 3. various buttons on the Puppet Dancer interface: the "reset" button erases demonstration data, the "training" button toggles between *authoring* and *playing* modes, and the large green and red buttons are the *approve* and *disapprove* reinforcement buttons, respectively.

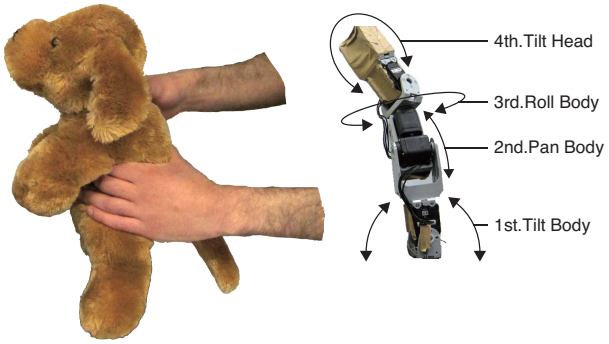

Figure 4: One of the stuffed toys and the exposed servo spine embedded within.

Figure 4 illustrates the spine's dimensions of movement: both the *leader* and *follower* can tilt their body forward and backward, pan from left to right (a side-to-side movement), turn or roll left and right, and tilt the neck up and down.

Both dancers are mounted rigidly facing each other on a plastic box containing the controller boards (Figure 1), serving as a platform for the *training* (mode-change), *reset*, and *reinforcement* buttons (**Figure 3**). The box connects via USB to a PC and the Puppet Dancer algorithm. Robot communication was accomplished using the C# Dynamixel library (www.forestmoon.com), and the Puppet Dancer algorithm was written in C# .Net 4.0. We implemented the buttons and button lights using an Arduino Uno.

Puppet Master Algorithm

The learning algorithm used by Puppet Dancer is an extension of the Puppet Master SBD algorithm [25, 26] to the dancing feature set and the needs of this project. First we will summarize the existing Puppet Master algorithm before detailing our adaptation.

The Puppet Master system, like Puppet Dancer, is an SBD platform for creating paired, style-oriented behaviors. Instead of dance, however, Puppet Master targets stylistic and interactive robot locomotion paths: the way a robot moves around a space to interact with a counterpart entity [25, 26]. Puppet Master also has both *authoring* and *playing* modes; however, with Puppet Master users cannot go back and forth between the modes: once initial demonstration is complete, training cannot be added.

Puppet Master is a pattern matching algorithm that runs at relatively high speed (40hz [25] and 15hz [26] depending on the previous implementation) to achieve interactivity, and uses heavy smoothing and frequency-analysis filtering to maintain coherent results. For *authoring*, users provided demonstration of a robot interacting with a person (locomotion path only, e.g., the robot stalking the person). Then, during the *playing* mode, Puppet Master searches the training set for locomotion paths similar to the real-time state, to inform ongoing robot movements. Thus, the robot output is a kind of patch-work of training data pieces (**Figure 5**). Training-data is searched every time step to incorporate ongoing real-time user input, and uses a window of data (1s) compared against a window moving over the training data set: this history is used to maintain movement coherency, e.g., the robot circling the person.

Here we explain the comparison metric. First, locomotion paths are distilled into defining scalar features: velocity, turning amount (left or right), relative look direction (one looking at the other), and relative position (e.g., behind and to the left of the other). These features form a vector at each time point that can be compared. The best-match training data is selected by minimizing the Euclidean distance between the real-time window of data and the moving window over the training data.

The best-match training cannot be copied verbatim to robot output as the features generally conflict, e.g., the required movements to match target velocity (from the best match) may contradict the relative position, or a robot may need to turn one way to move toward the target location, violating the target orientation. Puppet Master uses a complex compromising robot-movement generation method which balances the demands from the various features.

A major problem with the Puppet Master algorithm is movement jitter, where the robot will appear to vibrate rapidly in a way not consistent with the training data. This happens when the best-match source training data changes rapidly between conflicting source locations, when their match scores are similar; thus, rather than using coherent patches from the training data, in these instances only very small slices are used: this phenomenon is denoted in Figure 5. Puppet Master's solution was to a) heavily smooth the robot motions to filter the jitter and b) use frequency analysis to re-introduce the appropriate training movement detail that was lost in the filtering. This provided an improvement but results were not satisfactory to users [25].

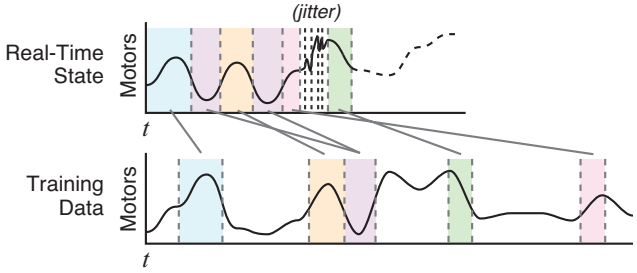

Figure 5. Real-time dance movement is a set of patches from the best-matching training data. Illustration only.

The summary provided here was for the purpose of explaining the background to the Puppet Dancer platform; important details were omitted for brevity and we recommend those wishing to implement this work to refer to the original Puppet Master papers.

Puppet Dancer Algorithm

The Puppet Master algorithm has important limitations which we needed to address for Puppet Dancer:

1. The locomotion-path feature set was fundamentally different from the dancing-robot body morphology.
2. There was no support for adding additional training, the user had to start over
3. The jitter was unacceptable for quality results.
4. There was no reinforcement learning ability to modify and improve generation quality with user feedback.

Feature Set and Output Generation

We selected the Puppet Dancer features as head tilt, body roll, body pan, and body tilt, where the combined features of both dancers form the data vector used for training-data search. These features map one-to-one to the physical motors. This means that, unlike with Puppet Master where the abstracted features were inherently intertwined (e.g., velocity, relative position, and relative orientation), Puppet Dancer features are all independent, for example, head tilt output can be solved without affecting body roll output. This greatly simplified the generation problem.

Output dance generation was done on each feature (i.e., each motor) independently. We could not set output motor locations directly from the training data as, for example, if a movement is intended to be slow (per the training) but the target is far from the current position, this would result in fast movements. Our solution is to apply delta movements (relative changes) from the training to the output motor's current position so that desired movements would be reproduced at the current motor location. For example, if the robot did a circle in data at one location, we could reproduce that circle at the current location. To maintain desired absolute position we applied weights: deltas moving away from the global target were dampened to 80%, while movements toward were amplified by 50%. This enabled the robot to tend toward the desired position while accurately performing the movements from the training data.

We ran Puppet Dancer at 40 Hz. This was selected through experimentation: with higher speeds we noticed no improvement, but with lower speeds we had a loss of interactivity.

Incrementally Adding Training Data

We enabled the incremental additions of training data (as the user moved between the *authoring* and *playing* modes) by storing new training sessions as entirely new data sets. For searching, all data sets were searched for the best match.

Managing Jitter

Puppet Master's jitter problem resulted when it rapidly jumped between dissimilar training data with similar best-match scores (**Figure 5**). Puppet Dancer's solution was to: a) encourage training data to be used in patches rather than single slices, and b) discourage repetitive jumping back and forth between similarly-scored training data.

For part a) we first tried an enforced minimum patch length, e.g., 0.5s. Unfortunately, this hindered the sense of Puppet Dancer's responsiveness when the *leader* made sudden movement changes, and so we implemented a hysteresis solution that we call sticky patches. Once a training region is selected as the best match, that region gets a bonus (100% of its given score) to subsequent best-match scoring such that Puppet Dancer tends to stick to that patch for generation, even if a slightly better match is found. A dramatically better patch with a better score can still overcome the sticky patch, maintaining high interactivity. This is illustrated in **Figure 6**: at points (a) and (b) the patch does not change even though a higher score was found elsewhere. The bonus is diminished linearly over 1.5s, so that quick patch changes are still possible without dramatically better best-matches.

Even with sticky patches, the repetitive oscillation between similarly-scored training patches problem still happened (albeit slower than before). We discourage this (part b of our solution) with a short-term black list. Once a training region is selected as best match to generate output, if Puppet Dancer decides to move away from that region to a better match, the original region is blacklisted for 2s (80 frames) so that it cannot be used again soon.

The combination of the above two methods was successful enough in fixing the jitter problem that we removed the heavy output smoothing and frequency-analysis filtering techniques required in the original Puppet Master.

Reinforcement Learning

To implement the reinforcement-learning interface mechanism, i.e., the *approve* and *disapprove* interface buttons, we weighted the training data. When one of the buttons are pressed, the last 1s (40 frames) training data used is applied a permanent weight: for approve, a 30% bonus is given, and for *disapprove*, a 30% penalty. The result is that approved regions are more likely to be used in the future, and disapproved regions are less likely.

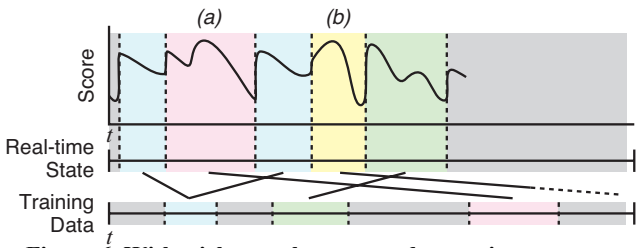

Figure 6. With sticky patches source data region may not change, even with a better match score (points a and b). Illustration only.

EVALUATION

The primary purpose of our evaluation was to investigate: a) is our implementation and algorithm successful in enabling easy to use end-user authoring of interactive robot dance? b) does our SBD interaction approach make sense to users? and c) how do users engage SBD interfaces?

We analyze implementation success in part by simply testing if participants can use Puppet Dancer to casually create custom interactive dances, without requiring training or assistance. We further compare our learning algorithm results to a behind-the-scenes experimenter remotely controlling the dance (Wizard of Oz technique): the "wizard" represents perhaps an ideal learning algorithm. The wizard observed the demonstration from the adjacent room using a hidden webcam, and remote controlled using a replica of the servo spine – manipulations were mirrored to the *follower* dog in real time.

We also specifically test the SBD reinforcement-learning mechanism, and in addition, explore if the *follower* robot needs to be interactive: we compare a non-interactive *follower* (simply dances without monitoring the *leader*) to our interactive version. We achieved this by having the dog replay random patches (2s long) from the training data with smoothed transitions; an approach taken from computer graphics texture synthesis [7].

Finally, we investigate overall appropriateness of our method: if our interaction design supports how participants actually want to create robot dances. We take a qualitative evaluation approach to study system use in comparison to expectations: for example, one early criticism of our work was that people will simply want a mirror behavior and do not want to customize their own. Ultimately, we aim to build understanding of how people use SBD, how they engage the demonstration task, and which methods they employ to work toward their final dance result.

Pilot Studies

We conducted several rounds of informal pilot studies to develop our evaluation method. One result was that we decided to have both an experimenter and the participant perform initial *authoring* together: an experimenter moved the *leader* while the participant used the *follower* to show their desired dance. Pilot studies showed that participants found it difficult early in the experiment to control both robots simultaneously, as the overall interactive SBD idea was new; later in the study participants perform simultaneous control. Further, the approach of having two participants (one for each the *leader* and *follower*) created

complications such as the *leader* evaluating results without knowing the *follower*'s intent. With an experimenter-controlled *leader*, participants focused well on the *follower* training, and this also improves *leader*-motion consistency across participants.

We also found that the beat that participants danced to would drift, particularly between the *authoring* and *playing* modes. The Puppet Dancer algorithm is sensitive to the timing of dance moves, and so this hindered the pattern matching and lowered the quality of the results. To solve this we added music to give the participants a consistent baseline for rhythmic movements.

Tasks and Procedure

The study consisted of a *comparison* phase that compared Puppet Dancer against other dance generation methods, a *feedback* phase that targeted the reinforcement learning mechanism, and an *open-ended* phase that explored how participants engage the overall training process when not given constraints (**Table 1**).

We first administered a demographics questionnaire, followed by a short (~2 min.) introduction to Puppet Dancer that focused on the SBD concept and included a demonstration of how to physically manipulate the robots.

For the *comparison* phase participants first authored a custom interactive robot dance by manipulating the *follower* dog while an experimenter moved the *leader* cat through a scripted and looped full-range-of-motion dance. Participants could demonstrate as much or as little as they felt was required to show their entire dance. During this phase the feedback, training and reset buttons were all hidden; participants verbally indicated to the experimenter when they were finished training. We used the song "Da Funk" by Daft Punk as background music. After demonstration participants were asked to control the *leader* to interact with the *follower* to evaluate the results for three cases: the a) Puppet Dancer algorithm, the b) remote-control wizard, and the c) non-interactive dance, order counter-balanced. Participants were not informed of the cases and were simply told that there were three different learning methods. We administered a short questionnaire after each case where participants gave open-ended feedback, rated the result using Likert-like scales, and rated their impression of the robot using the Godspeed anthropomorphism, animacy, likeability, and perceived Intelligence scales [3]. At the end of this phase participants ranked the three behaviors.

phase	variable	purpose
comparison	dance algorithm: Puppet Dancer, wizard, non-interactive	compare Puppet Dancer results to a person and to a non-interactive dance
feedback	feedback buttons enabled / disabled	explore how feedback buttons are used, if algorithm works
open-ended	none	explore freeform engagement of SBD and various mechanisms

Table 1. breakdown of the study organization

For the *feedback* phase participants were first introduced to the *approve* and *disapprove* buttons (we un-hid them at this point). They were asked to manipulate the *leader* to dance with the *follower*, and to use the buttons to improve the given behavior result. We used the same training data from the *comparison* phase and only the Puppet Dancer algorithm. The independent variable in this phase was the reinforcement method used, manipulated within subjects and order counter-balanced: either our reinforcement-learning method, or a variant where button presses were actually ignored, unbeknownst to the participant. This phase also used the song "Da Funk" by Daft Punk.

In the *open-ended* phase participants were introduced to the buttons and functionality that enabled them to add training incrementally or to "reset" learning to start over, and were given a chance to test the functionality to ensure they understood the controls. This phase was unstructured and participants were encouraged to just play with the system and use it as they wish to create a new interactive dance. They were asked to control both the *leader* and *follower* simultaneously, to aid the freeform nature of the phase, and the user feedback buttons were also enabled. During this phase we used a different song "Harder Better Faster Stronger" by Daft Punk, a change for variety. Participants were encouraged to use the system for as long as they like. Once finished, they completed a post-test questionnaire.

We recruited 11 participants from the general population in Tokyo, Japan, and each participant was paid 3000 JPY ($37 2011 USD) for their one hour participation; 4 female / 7 male, aged 20-52 (*M=29.0*). Some numbers presented in the analysis are lower than the total of 11 participants: in two cases, robot failure resulted in an early study finish, and in one case the video camera failed. Due to the reduced power of non-parametric statistics required to analyze Likert-like participant responses and behavior ranking [8], and given our small sample size, we have an increased risk of Type II errors (false-negatives). As such we present our data directly and discuss the overall trend in relation to our qualitative methods, and rely less on statistical analysis.

Comparison Phase Results

One concern of our work was that participants would simply want a *follower* which only mimicked the *leader's* movements or mirrored them; to investigate this we analyzed our video data to explore how people engaged the training. Of the 10 cases for which we had video, 2 participants were found creating a mimic behavior. The remaining 8 participants performed more complex, personalized dances which were interactive but did not directly mimic the *leader's* movements. Thus this supports

the idea that participants will have individualized and personal dances to teach the robot, rather than wanting mimicry only.

During the demonstration component, only one participant (who had a strong dancing background) commented that they wanted to also control the *leader* during *authoring* instead of the experimenter. This same participant also requested that the *follower* should learn how to dance appropriately to the mood of the music, e.g., differently for slow or energetic sections. No other participant commented on or was observed in the video data to alter their dance based on the music changes.

When participants manipulated the *leader* to test the *follower's* dance, some would make the *leader* cat dance in what appeared to be a free-form fashion, and themselves primarily appeared to be looking at the overall result. Five participants took this approach for Puppet Dancer, and two for the wizard and non-interactive cases. Two participants did this for all three. Other participants, however, would take a more deliberate leading approach where they would, for example, do a movement with the *leader*, and then stop to observe the *follower's* reaction, testing for the moves that they taught; these participants would commonly repeat a move and slow down their motions to be more deliberate, if they did not get their desired result. Three participants were observed doing this leading for Puppet Dancer generation, six for wizard and six for non-interactive. Five of these six participants overlapped for doing this for wizard and non-interactive, and the three Puppet Party leading participants used leading for all three. This leading behavior is particularly relevant to our Puppet Dancer algorithm, as a slowed-down deliberate leading motion does not match well with a more-natural motion as part of a dance: Puppet Dancer is highly time and speed sensitive.

Table 2 shows participant ratings for each algorithm in terms of how well they thought the *follower* learned the dance. Friedman's ANOVA found a significant difference between the rankings, $X^2(2)=4.923$, $p<0.05$, although post-hoc Wilcoxon Signed Ranks tests failed to reveal additional relationships; Friedman average rankings: Puppet Dancer *2.0*, wizard *2.36*, non-interactive *1.64*.

Table 3 shows how participants ranked the three behaviors in terms of their favorite result. Although the differences were not statistically significant, and no difference was found for how participants rated the three algorithms using the Godspeed questionnaires, this table suggests that the non-interactive was least liked. From our open coding of the video data, we noted that participants would, at times, exhibit frustration in their facial expressions and body language (e.g., by sighing and shaking their head). In the 10 cases for which we had video data, this happened with one participant for the Puppet Dancer case, 4 times for the non-interactive case, and was not observed for the wizard case. Overall, we believe that the above results lend support for the importance of having the robot's behavior to be real-time interactive, where generated movements are directly dependent on not only the robot's prior movements, but the ongoing movement set of the counterpart dancer.

	most negative				most positive
	1	2	3	4	5
Puppet Dancer	1	3	1	4	2
Wizard	0	3	1	4	3
Non-Interactive	1	6	1	2	1

Table 2. Frequency table of participant responses to "The robot learned the dance as expected," per behavior. 5 is most positive.

	1st	2nd	3rd
Puppet Dancer	5	3	3
wizard	5	3	3
non-interactive	1	5	5

Table 3. Frequency table of participant rankings of the three behavior types

Feedback Phase Results

Overall, we found no effect of the enabled / disabled condition on how participants used the buttons (e.g., frequency); it appeared as if they used the buttons the same way in both cases. Button use frequency depended on the participant, with number of hits ranging from 3 hits to 49 hits before the participant decided that they were done. One final observation was that all participants – save one – would hit a button once and watch the result before hitting again. The exception case would rapid fire the button until they got the desired result.

Participants rated the reinforcement-learning enabled condition as being more effective for improving the dance than the feedback-disabled condition, $T=2.5$, $p<0.05$, $r=-0.47$. There was no significant difference in how participants rated the robot on the Godspeed scales between conditions. When asked whether the like / dislike functions were useful, 3 disagreed, 1 was neutral, and 5 were positive. 75% (6) preferred the buttons on, and 25% (2) preferred the buttons off (1 did not answer).

Open-Ended Phase Results

Due to technical difficulties two participants did not complete this phase, and one participant was not video recorded. As such, only 8 participants are included in the video analysis.

We observed that participants engaged the interface in different ways to achieve the SBD goal of creating their own custom dance and enjoying the result. Three of the eight participants simply did a single demonstration attempt, and did not append training: two of these relied heavily on the reinforcement buttons to shape the result, while one user simply played with the result for an extended amount of time without trying to improve it.

In contrast, another two of the eight participants used the reset mechanism to start the dance over from scratch. One used the reset mechanism several (3) times to try different types of behaviors, playing with each result for a lengthy time; this participant did not use the reinforcement buttons to improve the dances. The other participant relied heavily on the reinforcement buttons to polish a dance, but at one point gave up and used the reset mechanism to re-try the same dance style from scratch.

Finally, the last three participants created their dances in parts using the training-addition mechanism. Two of these used the addition to build the dance up in small pieces (3 additions) that they each polished using reinforcement, i.e., to first train one particular dance move, perfect it using reinforcement, and then train an additional move.

No participant was found to use both the reset and addition mechanisms in their training and all participants but one used the reinforcement buttons. In addition, all participants but one were observed teaching the *follower* to interact directly with the *leader*'s movements, in contrast to the exception who simply moved the puppets in a seemingly non-interactive way. Finally, three participants would heavily lead the dance to test the result.

Post-Test Written Feedback Results

Most participants commented on the "fun" and "enjoyment" elements of being able to program the robot through demonstration, e.g., "*it was fun because I could program the robot to act as I like,*" and that the task of dancing was enjoyable: "*Because I was teaching how to dance, it was great and easy to use.*" In particular, several pointed out the importance of the attractive stuffed toys to the experience: "*because you used the animal plushies, the movements were cute and I had fun,*" and "*I was happy from the overall cute feeling.*" One participant related this enjoyment to the act of teaching: "*Being able to program my own robot is very useful. In addition, because you can customize it, it is easy to build personal attachment, and you see it as a friend you can communicate with.*" One person, however, noted that "*because of the mechanical [servo] sound, I could not help but perceive them as 'machines',*" and another said "*with the slow movement it was cute, with the quick movement it was a little scary and creepy.*"

Participants heavily commented on the usefulness of the various training capabilities, e.g., "*it was extremely good that I could use the training button to add more for the robot to learn,*" or that "*it was useful to be able to modify parts of the training using the like / dislike buttons.*" The reinforcement buttons in particular were praised for helping to build the learning scenario: "*in the case with having the like and dislike buttons I really felt that the robot is learning,*" and "*By using the like / dislike buttons I really felt like I could participate.*" Several participants noted the importance of exploration, e.g., "*from exploring the meaning of all the buttons, and from seeing the robot repeatedly respond to my directions, the feeling that the robot was learning emerged.*"

There was some confusion and negative feedback regarding the use of the buttons, however, and how it fit into the overall training system. For example, "*I could not properly confirm if, when I added training similar to previously existing training, if it overwrote that*" and one said "*I did not fully understand how to use the reset button so I finished without using it.*" One person highlighted the inability to undo an action, e.g., "*when [they] accidentally pushed the wrong like / dislike button.*"

There were only a few negative comments on the learning ability, e.g., "*there were also times when it moved differently than I expected,*" and one participant noted that, at "*times it moved better than I expected,*" and further, that "*it was great to teach these times using the like / dislike buttons.*"

There was very little feedback on the physical interface itself. One comment was that "*it would have been good to have the buttons attached to the stuffed animals*" rather than on the box.

Finally, one participant expressed confusion regarding the paired dance: "*more than thinking of how the cat [leader] should dance with the dog, I was thinking of how the dog [follower] should dance to the sound, or dance in a particular sequence*."

DISCUSSION

All participants engaged our interface, clearly understood the tasks with minimal instruction, and were able to author a wide range of interactive dances. Thus, our study supports our primary research questions, that a) our implementation and algorithm were successful in enabling easy to use end-user authoring of interactive robot dance, and b) the overall SBD interaction approach itself makes sense to users. Below we detail additional findings:

Individuals Engage SBD Differently – We were surprised at the wide range of SBD engagement styles that emerged from our study, and that participants used the tools provided to them in greatly different ways, for example, that some built behaviors piece-wise while others created them at once monolithically. No "average" user emerged from our study and there was no overarching theme or method to dance creation. This points to the importance of having a flexible SBD interface that supports various rapid prototyping and refinement methods to match individual preferences.

The Interactive Component is Important – Given that participants tended toward interactive dances in the free-form stage, and, following from the positive suggestions from our quantitative results, this shows the importance of having an interactive stylistic dance instead of a static pre-scripted dance.

People Want Fine-Resolution Behavior Shaping – Participants cited heavily the importance of the *approve* / *disapprove* reinforcement mechanism and used it often to refine behaviors. We believe that this points to the importance of providing SBD users with a fine-detailed and incremental way of modifying their behavior without resorting to the core demonstration mechanism.

People Test Results Using Leading Behavior – Participants used leading behavior, that is, repeatedly led the *leader* cat into moves to elicit the desired *follower* dog reaction, to test the learning algorithm. We think that perhaps this emerged as some participants may have believed they were intended to test the learning system. As we envision that end-users would rather play with the dance as an entertainment device, we recommend that future studies should directly attempt to minimize this effect.

FUTURE WORK AND LIMITATIONS

This paper is a preliminary step in the broader direction of SBD, and below we outline some directions for future investigation. Primarily, our implementation uses two simple 4-DOF robots and we need to consider more advanced robots such as humanoids, where the movement capabilities are at the same time more complex and more powerful. In addition to the need to adopt the Puppet Dancer algorithm to these more complex cases, there remains a difficult problem of performing the demonstration: which interfaces can we develop that simultaneously enable people to demonstrate their desired style while enforcing the movement capabilities and constraints of the robot?

We believe that the benefits of the SBD approach goes far beyond choreographing entertainment robots to dance and can be applied to authoring movements for more general-purpose robots. For example, a robot developed for a particular task such as waiting a home table can be shipped with a generic style programmed, with the understanding that end users could show the robot their particular tastes: this could greatly simplify the pre-programming required while at the same time improving versatility of the robot. With this in mind, future work needs to explore how to mesh hard-programmed task-oriented behaviors with SBD.

Through the process of designing and conducting this evaluation, and analyzing the results, we have spent considerable time contemplating how best to evaluate the quality of the Puppet Dancer algorithm generation results. In this project we decided to rely on participant reflection in a comparative study, but this remains an open question for future SBD work. Another example is that we used open-ended behavior dance, although it is not clear if this may have been confusing for participants, and we should explore if it would be more effective to provide clearer target dances.

In this paper we rely on the intuitive understanding of robotic behavior style, but do not formally explore what exactly style is. Future work will have to unpack the various variables which contribute to the impression of style, such as acceleration, hesitation, fluidity, etc. Further, we will have to look at the entire style envelope from motor responsiveness, to secondary motions such as the toy's floppy arms and ears, to the particular culture and context (or role of gender) within which the interaction takes place. Unpacking these variables will aid in developing clearer technical and evaluation targets, research methodologies, and ultimately more targeted and applicable results.

CONCLUSION

In this paper we presented Puppet Dancer, a *Style by Demonstration* platform for the creation of interactive, paired robot dance. We detailed our original algorithm, with several important extensions and improvements over prior work, and a formal qualitative study that demonstrated the success of Puppet Dancer and provided detailed insight into how people use *Style by Demonstration* in practice. Overall, we anticipate that Style by Demonstration will continue to emerge as an important domain in HRI, and that the interface design, algorithm, and detailed evaluation presented here will contribute to this field.

ACKNOWLEDGMENTS

We would like to thank our colleagues at the JST ERATO Igarashi Design Interface project for all their support and inspiration, and the anonymous reviewers for their helpful input. This research was supported by JST and NSERC research grants.

REFERENCES

1. Anon. Robot baby seals to replace pets in hospitals. CTV News Online, http://www.ctv.ca/CTVNews/-SciTech/20090112/robot_seals_090112/, The Canadian Press, Jan 12, 2009. (Accessed Oct. 20, 2010).
2. Anon. Forth valley royal hospital to use robot 'workers'. BBC News Online, http://-www.bbc.co.uk/news/10344849, June 17, 2010. (Accessed Oct. 20, 2010).
3. Bartneck, C., Kulic, D., Croft, E. and Zoghbi, S. Measurement instruments for the anthropomorphism, animacy, likability, perceived intelligence, and perceived safety of robots. *Int J Social Robotics*, 1(1):71–81, 2009.
4. Bartneck, C., Verbunt, M., Mubin, O., and Mahmud, A. A.. To kill a mockingbird robot. In *Proc HRI '07*, pages 81–87. ACM, 2007.
5. Bethel, C. L. and Murphy, R.. Non-facial and non-verbal affective expression for appearance-constrained

robots used in victim management. *Int J Behavioral Robotics*, 1(4):219–230, 2010.

6. Dillmann, R. Teaching and learning of robot tasks via observation of human performance. *Robotics and Autonomous Systems*, 47(2–3):109–116, June 2004.
7. Efros, A. A. and Freeman, B. Image quilting for texture synthesis and transfer. In *Proc SIGGRAPH '01*, 2001.
8. Field, A.. *Discovering Statistics through SPSS: (and sex and drugs and rock 'n' roll)*. Sage Publications, Thousand Oaks, CA, USA, third edition, 2009.
9. Frei, P., Su, V., Mikhak, B., and Ishii, H.. Curlybot: designing a new class of computational toys. In *CHI, 2000. CHI '00, The Hague, The Netherlands, April 1–6, 2000*, pages 129–136. ACM, 2000.
10. Gallaher, P. E.. Individual differences in nonverbal behavior: Dimensions of style. 63(1):133–145, 1992.
11. Gribovskaya, E. and Billard, A.. Combining dynamical systems control and programming by demonstration for teaching discrete bimanual coordination tasks to a humanoid robot. In *Proc HRI '08*, pages 33–40. ACM, 2008.
12. Halbert, D.. *Programming by Example*. PhD thesis, University of California Berkeley, 1984.
13. Harris, J. and Sharlin, E.. Exploring the affect of abstract motion in social human-robot interaction. In *ROMAN '11*. IEEE Computer Society, 2011.
14. Kanda, T., Kamashima, M., Imai, M., Ono, T., Sakamoto, D., Ishiguro, H., and Anzai, Y.. A humanoid robot that pretends to listen to route guidance from a human. *Autonomous Robots*, 22(1):87–100, Jan. 2007.
15. Kiesler, S. and Hinds, P.. Introduction to This Special Issue on Human-Robot Interaction. *Human Computer Interaction*, 19(1/2):1–8, 2004.
16. Lockerd, A. and Breazeal, C. L.. Tutelage and Socially Guided Robot Learning. In *IROS '04*, volume 4, pages 3475–3480. IEEE Computer Society, 2004.
17. Matsui, D., Minato, T., MacDorman, K. F., and Ishiguro, H.. Generating Natural Motion in an Android by Mapping Human Motion. In *IROS '05,* pages 1089–1096. IEEE Computer Society, 2005.
18. Michalowski, M. P., Sabanovic, S. and Kozima, H.. A dancing robot for rhythmic social interaction. In *Proc HRI '07*, pages 89–96. ACM, 2007.
19. Otero, N., Alissandrakis, A., Dautenhahn, K., Nehaniv, C., Syrdal, D. S., and Koay, K. L.. Human to robot demonstrations of routine home tasks: exploring the role of the robot's feedback. In *Proc HRI '08*, pages 177–184. ACM, 2008.
20. Raffle, H. S., Parkes, A. J., and Ishii, H.. Topobo: a constructive assembly system with kinetic memory. In *CHI '04*, pages 647–654. ACM, 2004.
21. Saerbeck, M. and Bartneck, C.. Perception of affect elicited by robot motion. In *Proc HRI '10*, pages 53–60. ACM, 2010.
22. Sung, J.-Y., Grinter, R.E., Christensen, H. I.. "pimp my Roomba": designing for personalization. In *CHI '09*, pages 193–196. ACM, 2009.
23. Sung, J.-Y., Guo, L., Grinter, R. E., and Christensen, H. I.. "my Roomba is Rambo": Intimate home appliances. In *UBICOMP '07*, volume 4717/2007 of *Lecture Notes in Computer Science*, pages 145–162. Springer-Verlag, Berlin, New York, Heidelberg, 2007.
24. Young, J. E.. *Exploring Social Interaction Between Robots and People*. PhD thesis, University of Calgary, Calgary, Canada, August 2010.
25. Young, J. E., Igarashi, T., and Sharlin, E.. Puppet master: Designing reactive character behavior by demonstration. In *SCA '08*, pages 183–191. Eurographics Association Press, 2008.
26. Young, J. E., Ishii, K., Igarashi, T., and Sharlin, E.. Style-by-demonstration: Teaching Interactive Movement Style to Robots. In *ACM IUI '12*. ACM, 2012.
27. Young, J. E., Sung, J.-Y., Voida, A., Shrlin, E., Igarashi, T., Christensen, H., and Grinter, R.. Evaluating human-robot interaction: Focusing on the holistic interaction experience. *Int J Social Robotics*, 3(1):53–67, 2010.