Design and Evaluation Techniques for Authoring Interactive and Stylistic Behaviors

James E. Young, University of Manitoba
Takeo Igarashi, The University of Tokyo / JST ERATO
Ehud Sharlin, University of Calgary
Daisuke Sakamoto, The University of Tokyo / JST ERATO
Jeffrey Allen, JST ERATO

We present a series of projects for end-user authoring of interactive robotic behaviors, with a particular focus on the *style* of those behaviors: we call this approach Style-by-Demonstration (SBD). We provide an overview introduction of three different SBD platforms: SBD for animated character interactive locomotion paths, SBD for interactive robot locomotion paths, and SBD for interactive robot dance. The primary contribution of this paper is a detailed cross-project SBD analysis of the interaction designs and evaluation approaches employed, with the goal of providing general guidelines stemming from our experiences, for both developing and evaluating SBD systems. In addition, we provide the first full account of our Puppet Master SBD algorithm, with an explanation of how it evolved through the projects.

Categories and Subject Descriptors: H.5.2 [**Information Systems**]: Information Interfaces and Presentation: User Interfaces - Input Devices and Strategies

General Terms: Design, Algorithms, Human Factors

Additional Key Words and Phrases: Style-by-Demonstration, Human-Robot Interaction, Human-Computer Interaction, Programming-by-Demonstration

ACM Reference Format:
(to be completed) *ACM Trans. Embedd. Comput. Syst.* 0, 0, Article 0 (0), 38 pages.
DOI: http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

We use the word *style* to mean "*how* something is done...the way in which a behavior is performed" [Gallaher 1992]. The style layer of a behavior can be quite separate from the particular task or action which a robot may be performing: behaviors such as walking has styles that vary between people, and even vary for a given individual depending on their mood or the social context. Literature in social psychology highlights the important role of style (or *expressive behavior*) on how people are perceived, and its implications on inter-personal communication and social interactions (e. g., see [Ambady and Rosenthal 1992; Butler et al. 2003; Gallaher 1992; Hall and Coats 2005]). For example, it has been shown that the style of actions impacts perceptions of hierarchy [Hall and Coats 2005], and that suppressing stylistic (expressive) actions can disrupt interactions and induce stress in others [Butler et al. 2003].

Thus if we accept that people tend to anthropomorphize robots in their everyday spaces and in some regards treat them as social entities (see, e. g., [Bartneck and Forlizzi 2004; Dautenhahn 2002; Forlizzi and DiSalvo 2006; Kiesler and Hinds 2004; Nass and Moon 2000; Reeves and Nass 1996; Sung et al. 2007; Young et al. 2009]), then we can expect that style will be important for human-robot interactions in similar ways as it is for human-human interactions. It is important for the field of Human-Robot Interaction (HRI) to consider how a robot's expressive behavior, its style, is perceived by people and how it impacts the spaces the robot occupies.

Gallaher [1992] highlights how behavior style is extremely individual and varies widely from person to person. Therefore, we posit that it will be useful if people can customize their robots' behavior styles to suit their culture and specific tastes, for example, similar to how a business mentor can teach a junior how to improve their sales pitch or give a confident handshake. One approach toward this goal is to use Programming by Demonstration (PBD), where end users can *program* their robots simply by providing a *demonstration* of the desired behavior [Dillmann 2004; Halbert 1984]. Most PBD work to date focuses on a task goal only – *what* is to be done – and ignores the expressive style – *how* it is to be done. New algorithms and interaction techniques need to be developed to enable PBD to target the style component of interactive behaviors. We call this approach Style-by-Demonstration (SBD).

The goal of our work presented in this paper is to develop understanding of how people may use, engage, and interact with SBD systems. Given this goal, we aimed to build complete interactive systems that people could engage rather than to focus on more narrow questions of perceptions and definitions of style, for example, considering robot shape or methodologically exploring motion attributes such as acceleration or hesitance. Our particular work focuses on how the fine-detail aspects of motion (the high-frequency component, or *motion texture*) relates to the perceived style.

We designed, developed, and evaluated three original SBD projects that we bring together in this paper for meta-analysis: SBD for animated character interactive locomotion paths [Young et al. 2008], SBD for interactive robot locomotion paths [Young et al. 2012], and SBD for interactive robot dance ([Allen et al. 2012], Fig. 1). As part of this we iteratively developed an SBD algorithm we call Puppet Master. Our approach with these projects has two key features which clearly differentiate them from prior PBD work: we focus on the low-level style of robotic movements and do not address other aspects of style, and, our learned behaviors are *interactive*, where the robot learns how to interact in real time to changing and unpredictable human input.

Throughout conducting these projects we faced many challenges related to effective SBD interaction design and to the question of how to evaluate SBD platforms. In this paper we present an overview of these SBD challenges and a detailed analysis of the

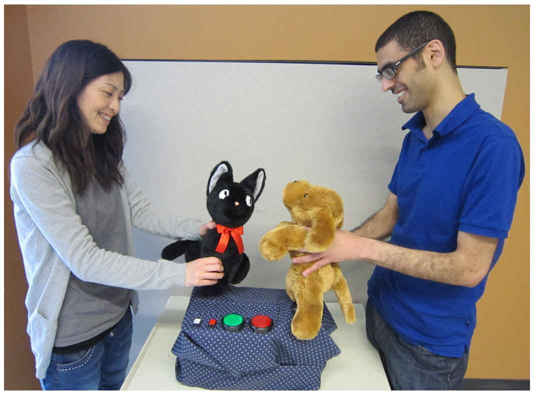


Fig. 1: SBD for authoring interactive robot dance. The *reactor* dog is learning to dance with the *leader* cat.

various approaches employed across the different projects. Further, we present for the first time the final (after iterations) technical learning algorithm used in these projects. Our contributions are:

(1) a complete account of the Puppet Master SBD algorithm after evolution through several projects.
(2) an overview of three original SBD projects and implementations.
(3) an analysis of SBD interaction design challenges and approaches.
(4) an analysis of SBD evaluation challenges and approaches.

2. RELATED WORK

A great deal of work aims to create life-like and convincing interactive behaviors. One common approach is through explicitly programming the behavior model (e. g., as with [Blumberg and Galyean 1995; Breazeal and Scassellati 1999; Maes 1995; Reynolds 1987]), where the programmer defines what to do for particular situations. The idea of programming such behaviors with an explicit focus on style has been prevalent in social HRI and "Natural HRI," well exemplified, for example, by work which explores a robot's appropriate use of human-like social cues such as gaze [Mutlu et al. 2009], visual attention [Staudte and Crocker 2009] or head-nodding [Sidner et al. 2006], appropriately following social norms such as not "cheating" [Short et al. 2010] or proxemics-based behavior [Gockley et al. 2006]. Other examples include robots attempting to have their "presence" felt appropriately [Hüttenrauch and Eklundh 2004], or investigations on how to engage [Michalowski et al. 2007] or calm [Bethel and Murphy 2010] a person by how it moves, or perhaps even the entire field of android science (e. g., [Sakamoto et al. 2007]) which aims to make robots that pass for humans [Ishiguro 2007]. Such projects do not aim to have their behaviors customizable by users and thus are generally (and often with great difficulty) hard-coded for the specific projects.

PBD has been successfully employed since the early days of robotics [Halbert 1984] for such applications as learning specific navigation routes [Kanda et al. 2007] or physical tasks [Gribovskaya and Billard 2008; Otero et al. 2008]. Many of these systems explicitly aim to identify key points and smooth between them [Vakanski et al. 2012] to remove, for example, "human inconsistency problems" [Aleotti et al. 2005]; while this is important for trajectory planning, and such trajectories inevitably contain stylistic elements related to the demonstration, these techniques also remove important style-oriented detail. Notable exceptions include robots which learn human-like motions and poses [Matsubara et al. 2010; Matsui et al. 2005] or stylistic movements [Frei et al. 2000; Raffle et al. 2004], although these provide only a static replay of demonstration verbatim and the resulting behaviors are not interactive, or a system for learning interactive eye gaze patterns [Mohammad et al. 2010], although this requires a pre-processed database of examples. Other PBD robot projects use style and emotion-charged elements as part of the demonstration-task interaction support: Breazeal et al.'s Leonardo robot uses facial expressions and style-laden gestures, while being taught, to convey such messages as lack of understanding or surprise [Breazeal et al. 2004; Lockerd and Breazeal 2004]. Here the stylistic motions are not learned but serve as communication tools; the tasks being learned are goal oriented.

PBD has also been used in the animation and interaction communities. Early PBD was used to automate GUI operations [Cypher 1991; Maulsby et al. 1989] or specifically define character planning models [Dinerstein et al. 2007]. For example, Pavlov [Wolber 1997] defines the low-level stimulus-response behavior of interactive agents. These systems define behavior using logical sequences of goals, the *what* to do, and do not provide tools to enable the user to tell the agent the style of *how* to do it. Further, although there is a great deal of work for defining the style properties of animation (e. g., [Dontcheva et al. 2003; Heloir et al. 2008; Hertzmann et al. 2002; Igarashi et al. 2005; Thorne et al. 2004; Torresani et al. 2006]), these focus on static playback of demonstrations and do not generalize the result to be interactive in real time to another entity. While some animation work is interactive to user input, these generally require a large, pre-processed (by an expert) database, and requires the mapping from user input to output to be explicitly (and often tediously) defined by the programmer [Lee and Lee 2004; Lerner et al. 2007; Wiley and Hahn 1997]. Further, these systems primarily target physical motion plausibility (punch, jump, collision avoidance, etc.) and not behavior style.

Robot customization has been explored beyond behavior, for example, with the robots' 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 [Sung et al. 2009b, 2007]. Thus we believe that the kinds of customization enabled by SBD will be of interest to end users.

People attribute style, personality, and emotions to even simple movements of abstract shapes [Heider and Simmel 1944; Kassin 1982; Scholl and Tremoulet 2000; Tremoulet and Feldman 2000], and the style of movements with animated characters has been touted as being critically important for how the character is perceived [Thomas and Johnston 1981]. Therefore we expect people to attribute style to all robotic movements, even those not explicitly designed, and that this style will impact interaction: SBD leverages this channel of communication. Style has been successfully leveraged for conveying behavior character, both in animation and research, for example making scripted animation actions such as "pick up a glass" to be "neutral," "shy," or "angry" by altering movement style [Amaya et al. 1996], or more general methods for applying style to human motions such as gestures [Torresani et al. 2006], and so we expect similar methods to be relevant for robots. Although there are few interactive-style specific robot projects to date (e.g., [Harris and Sharlin 2011; Saerbeck and Bartneck 2010]), a great deal of HRI work has a style element, for example, the consideration of robotic proximity during interaction: while many target practical mobility and vision challenges (e. g., [Byers and Jenkins 2008; Chueh et al. 2006; Liem et al. 2008]), others look at how close a robot should be to a person for comfort [Yamaoka et al. 2008], or how a robot should follow naturally [Gockley et al. 2007] (i. e., copying a path versus shortest route). Our work continues this research direction by enabling people to teach robots how to interact in a desired style.

Some work aims to simplify the generation of stylistic behaviors, for example, through leveraging artistic systems such as Laban Effort Analysis [LaViers and Egerstedt 2012] (for dance styles in this case). Like many other systems mentioned above that attempt to automatically generate style (e.g., [Heloir et al. 2008; Matsubara et al. 2010; Torresani et al. 2006]), these approaches often clearly define (what they mean by) style and provides means to generate it, techniques which will be important for continued work in this area. Our current work does not directly build on these methods, in part because most of them do not target *interactive* style, require expensive pre-processing, or target specific stylistic needs instead of being general purpose, and in part since the goal of our work was not to further unpack what exactly *style* is or how it can be represented. However, continued work in this area could benefit from reconsidering the feature space used (see Sec. 3) for developing robust SBD algorithms.

Due to the task focus of most robotic PBD, evaluation has generally been technical and goal-oriented, measuring accuracy or task-completion time (e. g., [Breazeal 2002; Matsui et al. 2005]) – these methods do not directly apply to the more subjective SBD results. Further, there is increasing evidence that human-centric and social aspects of HRI, such as is core to SBD, are particularly prominent in interaction and must considered in evaluation [Bartneck et al. 2007; Short et al. 2010; Young et al. 2009, 2010].

The quantitative method alone of distilling these complex human-oriented aspects into a set of statistical numbers is often insufficient for properly describing the kinds of subjective interaction targeted by this kind of work [Strauss and Corbin 1998], and so qualitative and exploratory evaluation methods often serve as the primary element of otherwise controlled studies (e. g., [Forlizzi and DiSalvo 2006; Sung et al. 2009a, 2007]), where the methods are used to describe interaction and to construct grounded interaction theories. Although less common for PBD, existing qualitative evaluations explore and describe interaction experience as a way of building understanding of how PBD can be employed, for example, integrated into educational tasks [Frei et al. 2000; Raffle et al. 2004]. We follow this precedent of using exploratory, interaction-experience oriented methods to evaluate SBD.

3. THE PUPPET MASTER ALGORITHM

We developed three SBD projects and their interaction and interface designs: authoring the interactive path of an animated character, interactive locomotion path of a disc robot, and interactive robot dance; we refer to these shorthand as the *animation*, *robot follower*, and *puppet dancer* projects. To realize these projects we developed our Puppet Master algorithm for SBD, which started as an initial animation-only technique [Young et al. 2008] but was altered and improved significantly for the robotic projects; the entire algorithm and cross-incarnation improvements has not before been presented in detail. Full details on the interfaces are presented later in the paper.

In our setup SBD requires two distinct phases: *authoring*, where the user provides a demonstration to the robot of how they would like it to interact, and *generation*, where the robot exhibits the interactive behavior as learned. Further, we focus on enabling rapid and iterative behavior creation, where the user can instantly see the results of their demonstration, which places limitations on the interfaces and learning algorithms used. As such, we design for authoring sessions to be from 30 s to roughly 3 m, to enable rapid end-user prototyping.

An important but perhaps nebulous point of our SBD systems is that authoring requires two entities moving simultaneously: a *leader*, which moves freely and does not learn from the demonstration, and a *reactor*, which monitors the leader's actions and exhibits an interactive stylistic behavior in response. Without an exemplar of the leader's movements during demonstration the reactor does not have a reference point from which to learn the interactive component of the demonstration: it would only be able to reproduce the static path. This is in contrast to many PBD systems where a user only has to demonstrate how the robot should move. Our SBD process is as follows: for authoring, the leader performs an exemplar of how it will generally move, and the reactor is given a demonstration of how it should interact with the leader's movements. For generation, the leader's movements are unconstrained and the reactor's movements are generated to interact with the leader using the demonstrated behavior style (Tab. I). In all cases our reactor does not start with a pre-programmed goal or behavior (e. g., to follow); the entire behavior is learned from a short authoring. In retrospect, our particular paired-motion setup has a limitation in that it does not address the realistic case of how interactive learning is often reciprocal: a leader may adapt to a learner and the roles may shift fluidly, for example, with a learner leading to test what they learned. We return to this point in our analysis later in the paper.

An important point to make here is that we view SBD authoring as an engaged and complex real-time two-entity acting task, and not a piecewise series of stimuli and responses [Pavlov 1927] – thus we do not use the stimulus-response vocabulary to discuss our work. In the stimulus-response paradigm the user consciously chooses stimulus, provides an example of that stimulus, and demonstrates an appropriate response.

Table I: the leader and reactor roles in each SBD phase

	phase	
	authoring	*generation*
leader	moves in an exemplary way to represent the final target scenario, gives the reactor something to interact with	moves naturally however they wish and fully expects the reactor to interact appropriately
reactor	is shown how to interact with the leader in the appropriate style, dynamically accommodating the leader's changing movements	automatically interacts with the leader in real time using the behavior style demonstrated during *authoring*

For our projects the user simply provides a real-time two entity acted out higher-level example of the behavior without necessarily thinking on the stimulus-response level.

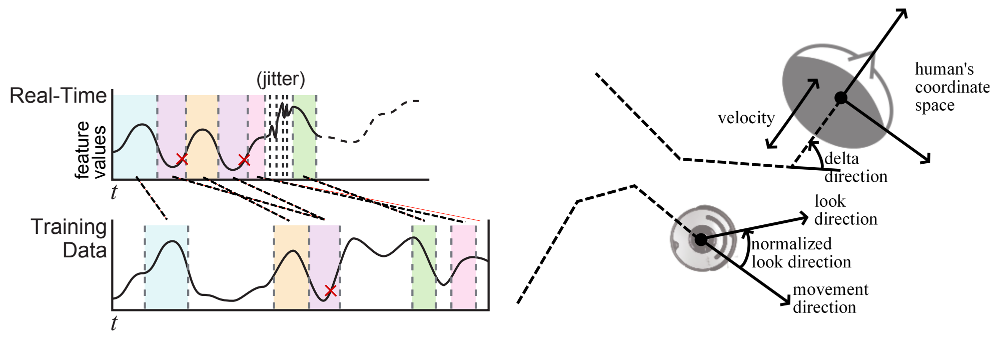
Puppet Master is a pattern matching algorithm with its roots in the Image Analogies project [Hertzmann et al. 2001]. It generates output at 40 Hz to achieve interactivity; 40 Hz was informally chosen through experimentation to minimize processing requirements while maximizing interactivity.[1]. During authoring, Puppet Master only records the combined features of the leader's and reactor's movements, and processes them in real-time into an appropriate feature set. For every cycle (40 Hz) of generation, Puppet Master searches the training data for interactions similar to the real-time state to inform the next generation output, and uses frequency-analysis and hysteresis filtering to maintain coherent results. Thus, Puppet Master's generated output is a processed patch-work of pieces from the training data (Fig. 2(a)).

Below we detail the feature-selection, the search and the output generation and filtering mechanisms. During this section, it is important to keep in mind that Puppet Master was created for enabling SBD interaction sessions that could be studied. As such, some of the algorithmic choices and parameters made were selected informally through use and testing and were often not robustly fine-tuned and optimized.

3.1. Features

Puppet Master requires a set of relevant scalar features to be extracted from movement data for its search (e. g., as in Fig. 2(b)). The features used are selected based on

[1] 15 Hz for robot follower to accommodate the slow response time of the robot and the problem of jitter, explained below; improvements in the algorithm since, also below, makes this accommodation unnecessary.



(a) Real-time Puppet Master generation is a processed set of patches from the best-matched training data. Red x's denote locations of auxiliary actions (robot sounds). Illustration only.

(b) Some of the data features used in our implementations. All features except relative position are on both entities, but only shown on one for image clarity.

Fig. 2: Core components of the Puppet Master algorithm.

the target platform to represent the defining aspects of the motion style. We address specific features for our projects later, but for now highlight that for Puppet Master they must be scalar values (in comparison to, e.g., descriptive ones) and must be carefully selected to represent the aspects of motion one is trying to capture.

3.2. Best-Match Searching

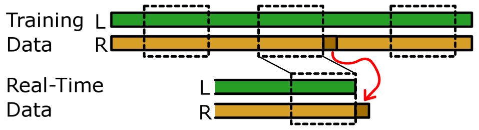
We search the training database for the leader and reactor configuration most similar to the current run-time configuration (based on the features used), and use this *best match* to inform the next generation step. We use a window of data (1s) compared against a moving window over the training data (Fig. 3). This neighborhood search enables us to capture the derivatives of the actions, such as one entity moving toward another or circling, or a dancing robot starting a dip or twist. The 1s window size was selected through testing as a balance between interactivity and quality, where larger window sizes created a delay before appropriate interaction was matched (even when history was weighted to favor recent data linearly or with a Gaussian distribution), and shorter window sizes made longer interactions such as one entity circling another (for the following robot) fail to be matched.

The comparison metric we use is Euclidean distance. At any given time point the scalar features can be used to form a multi-dimensional vector, such that the Euclidean distance between a given training-data vector and real-time-data vector can be easily calculated. Lower values represent a better match, and to compare windows we sum the distance of corresponding vectors. (We use Euclidean distance squared to save computation as we are simply minimizing the result). This process is a brute-force method for finding the nearest neighbor in our feature space over the window.

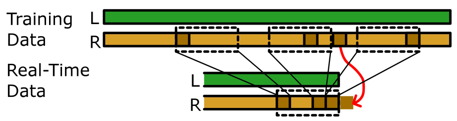
3.2.1. Similarity and Coherence Match. The searching is split into two simultaneous components, a similarity match and a generation coherence match (inspired by the Image Analogies technique [Hertzmann et al. 2001, 2002]), and a balancing algorithm to combine the two. The similarity match focuses on the relationship between the entities: it compares a window of the most recent real-time situation data – both the generated reactor and leader movements – to a moving window over the training data (Fig. 3(a)).

The coherency match emphasizes the similarity between generated reactor movement and the demonstration, and puts less emphasis on the leader; it uses the features of the reactor only in calculating similarity.[2] Coherency match does not search the entire training set, only regions which were recently used (within the current window size) to generate the output (Fig. 3(b)); the algorithm finds the source region for every recent output generated, and compares that region to the recent reactor movement to finds the region that is most similar. The intuition is that when situation similarity

[2]The exception is for interactive locomotion paths where we use the relative-location feature: for example, for a reactor to finish a circle around the leader it needs to track the leader



(a) For similarity search both the leader's and reactor's data are used to determine which training data to use for the next generation.



(b) For coherency search if the similarity score is low we attempt to continue a previous training region already used in generation

Fig. 3: Searches happen using a 1 s window of training data. In both searches the training data with the best feature match is chosen for generation.

match is weak (i. e., no training data matches the current real-time situation), generation coherency continues a previously-used match to ensure that the reactor is at least moving in a way that was demonstrated.

3.2.2. Similarity and Coherence Balancing. The results of the simultaneous searches must be combined or we must alternate between the two. The work we based this approach on [Hertzmann et al. 2001] simply selects the better score from the two after applying a static weighting bias k. This does not work with real-time data as: if coherence match is used to generate output for several consecutive steps, then the reactor generation will be increasingly similar to the reactor demonstration – irrespective of the leader. Coherence match continues to improve until it ultimately is used exclusively, with the result not considering the leader's actions. We call this problem *coherence loops*.

This problem did not occur in the original texture synthesis application as all data was available at the beginning and multi-resolution approaches ensured global coherency [Hertzmann et al. 2001]. We cannot use this given our real-time constraints and the lack of a lookahead method. Our solution is to define a target similarity-to-coherence use ratio (we used 1:1) and to dynamically tune k to encourage our target ratio; we use a 1:1 ratio and modify k by 5% each step. A similar algorithm is used in texture synthesis systems to match the overall color histogram [Kopf et al. 2007]. We did not observe a large difference in result with different k target or tuning step sizes; the key seems to be simply preventing one matching technique from dominating and keeping both matching regularly. Once the appropriate training data is selected the data immediately following it is used as the best match (Fig. 3).

3.2.3. Hysteresis Smoothing. One problem with the matching and balancing approach is noise. Instability in the similarity metrics (jumping between regions) and switching rapidly between similarity and coherence matches can cause large, rapid variations in the source data passed to the generation system, resulting in distracting rapid generation movements not consistent with the authoring. This phenomenon is highlighted in Fig. 2(a), page 6. We use hysteresis smoothing to discourage rapid jumps between source regions in the training data and to encourage patches (as in Fig. 2(a)). Intuitively this can be thought of as sticking to a region, where it is easier to enter a training region than it is to leave it.

Once a training region is selected we give the subsequent region a temporary bonus to scoring, encouraging its use such that Puppet Master will tend to stick to that patch for generation even if a slightly better match is found. A dramatically better match will still overcome the hysteresis to maintain high interactivity. As a patch is continued the bonus diminishes linearly (over 1.5 s) to ensure that other training data can be used where appropriate. With this alone Puppet Master would cycle between similar patches at a period roughly the same as the hysteresis decay (1.5 s in this case). We discourage this by black-listing used training regions for 2 s such that they cannot be used during that time. The result of this smoothing is passed to the output module to generate the next robot movement. Both parameters (1.5 s hysteresis decay and 2 s blacklist) were chosen simply as values significantly larger than the window size such that a) a recent match would maintain a bonus even after the window has moved past it a little (to aid in continuity) and, b) a region is black listed long enough for another pattern (based on another region) to start, thus making the black list unnecessary.

3.3. Output Generation

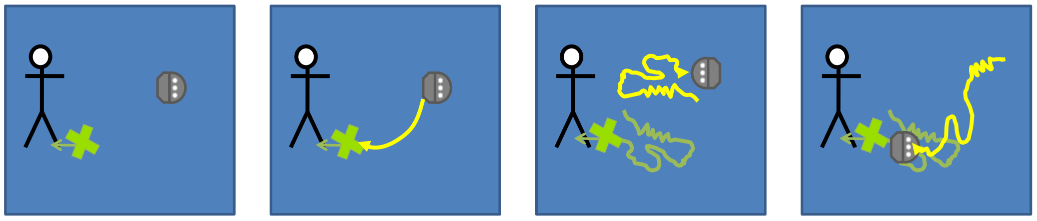
Best-match training cannot be copied verbatim to output. First, the feature set may not map directly to robot output causing ambiguities: for example, the best-match's target forward velocity may contradict the target relative position which requires a negative velocity. Any solution must be feature-set specific and must sacrifice some features for favoring others. Our general solution is to solve for target state (on each independent output), regardless of velocity, then scale that movement to the target velocity. This maintains the authored speed of the robot while still moving toward the target.

The second problem is that the robot movement detail may contradict the overall movement target. For example, the target may have the robot bop left while leaning right: if the robot is not already leaning right, does it move left to perform the bop motion or move right to reach the global target? Our generation approach, a key insight of Puppet Master, is to decompose the motion into its low-frequency (move to global position) and high-frequency (texture or detail) parts and then balance them, as a means of integrating both the global target and movement detail into each movement.

3.3.1. Frequency-Analysis Output Blending. The algorithm is as follows; we use the robot follower platform for illustration (Fig. 4). First, given an absolute target (Fig. 4(a)) we solve for the robot command that will move it to a given Puppet Master target state (Fig. 4(b)). This is the low-frequency component of the desired movement as it changes relatively slowly over time, which places the robot in the global position. However, as the robot cannot instantly keep up with the moving target over time, the texture (or details) of the desired movement are lost. Second, we find how the robot can reproduce the exact desired path (Fig. 4(c)), by solving for the delta movement rather than the target location. This maintains the high-frequency component, but drift means that the robot soon loses its relative localization: the low frequency component.

We combine the above two components by using a weighted average of the two resulting robot commands (Fig. 4(d)). Focusing on the high-frequency component results in better texture retention but more location drift, and a focus on the low-frequency component has the opposite effect. We use a 70/30 high/low-frequency balance to emphasize texture and to be looser on relative robot location; we found any balance strongly in the high-frequency component's favor to yield good results and selected this value informally through experimentation. The intuition is that existing high-frequency robot movements which tend to move away from the target location are slightly modified by the algorithm to correct their direction, while movements that are already directed toward the overall target are generally unchanged. The overall input, training, and generation data-flow and process is given in Fig. 5.

We believe that our solution is a unique method for filtering a desired movement against robot capabilities while focusing on the balance of texture and global motion, and can be used for the general problem of applying less-constrained movement behaviors (e. g., as demonstrated by a human that may not match exact robot capabilities) to the constraints of particular robots. That is, if a robot system is given an unpredictable movement path without look-ahead capabilities, it can apply our method to adopt the



(a) robot and target state

(b) solve for robot to reach target in one time step if possible

(c) target state over time forms path, solve to reproduce texture

(d) combine movements, tend toward target over time

Fig. 4: Our frequency-analysis robot movement solution.

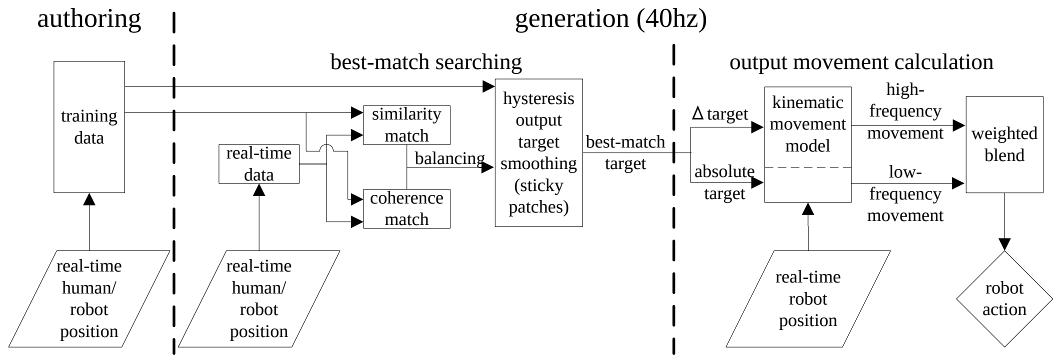


Fig. 5: Puppet Master's process and components.

path while focusing on maintaining style if it has a kinematic model which enables it to: a) calculate the closest possible fit to reproducing that path and b) calculate the shortest-path route to a far absolute target.

3.4. Minor Puppet Master Extensions

Here we provide details of algorithm extensions used throughout our projects.

3.4.1. Incrementally Adding Training Data. Data can be added incrementally over several training sessions simply by storing each session as its own training set. Then, for searching, all data sets are searched to find the best match. This is used in the robot dancer project (Sec. 4.3).

3.4.2. Simple Reinforcement Learning. It is possible to incorporate user feedback to fine-tune the algorithm in real time, for example, by the user indicating good or bad results. We implemented rudimentary reinforcement learning by adding added positive or negative weights. This was used in the puppet dancer project (Sec. 4.3).

3.4.3. Auxiliary Actions. The following extension is not addressed in the discussion in the paper but is included here for completeness; use cases and evaluation results can be found in prior work [Young et al. 2012]. It is possible to demonstrate discrete actions in tandem to the Puppet Master demonstrations (e.g., when to smile, take a picture, etc.). This can be achieved technically by associating the demonstrated actions with the training data on the time axis, such that the data is marked with the action at the appropriate time. As marked data is used during generation the associated actions are performed; this is illustrated in Fig. 2(a), page 6.

3.5. Specific Features Used

Below we present the features used for projects introduced in this paper, but these will have to be selected and tested anew for each use case.

3.5.1. Features for the Animation and Robot Follower's Interactive Locomotion Paths. We considered the important features for interactive locomotion to be the details of the reactor's path, and the relative movements to the leader's location; we do not use global features such as reactor and leader location in the space. We distilled the locomotion paths into the following features (Fig. 2(b), page 6):

Velocity. Captures speed and acceleration-related aspects of behavior such as different reactions for the person being stopped, accelerating, fast, or slow.

Relative position. Reactor's position in relation to the leader's position and look direction (coordinate space). This captures such relational behavior as following, circling, and approaching.

Normalized look direction. Look direction relative to movement direction, to capture, e. g., if the leader or reactor is backing up or moving sideways.

Relative look direction. The difference between the look directions, e. g., to capture when the leader and reactor are facing each other or looking away.

ΔDirection. Change in direction, to represents locomotion path detail, e. g., to match movement styles such as shaky or smooth.

All features except relative position are used for both the leader and the reactor's similarity calculations. As our generation is from the reactor's perspective we exclude considering the leader's perspective directly. We explored many features not presented here (e. g., delta distance between the characters) but generally found that the feature was already captured in the system and could be reduced to one of the ones above.

3.5.2. Features for Interactive Robot Dance. For robot dance we envision the reactor robot performing body poses and movements in relation to the leader, for example, to capture one robot leaning to the side while the other does a turn. Thus, as features we used the body pose of the robots themselves; in this case the robot's have a four degree-of-freedom spine (see Fig. 10(b), page 19). The features used are the head tilt, body roll, body pan, and body tilt, considered simultaneously for both the leader and reactor.

3.5.3. Frequency Analysis on Specific Features. Our frequency-analysis output technique is dependent on the specific movement constraints of each robot. For the robot dancer, as the features used (head tilt, body roll, body pan, body tilt) matched the motor output capabilities exactly (Fig. 10(b), page 19), solving for the outputs was trivial. For the robot follower platform, we used a kinematic model of the robot that describes its movement capabilities and constraints to generate the frequency components above, with commands guaranteed to be producible by the robot. We solve for robot movement commands (velocity and turning radius in this case) for the high and low frequency movements: given a target $\Delta x, \Delta y, \Delta\theta$ ($theta$ is look direction) simple trigonometry gives us our turning radius ($r = \Delta x + \frac{\Delta y}{\tan\theta}$) and velocity ($vt = \theta r$, t is time), Fig. 6.

3.6. Evolution of the Puppet Master Algorithm

The Puppet Master algorithm as presented above is the final result of iteration and evolution through the three projects presented in this paper. Here we briefly discuss

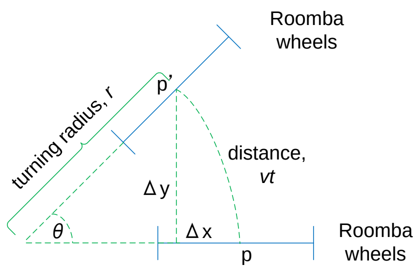


Fig. 6: Movement capabilities of our follower robot, an iRobot Roomba, showing the relationship between its control scheme of *velocity* (distance / time) and *turning radius* and the resulting $\Delta x, \Delta y$.

the main changes and reasons behind them. A major problem with the original animation Puppet Master was generation jitter when the algorithm confidence was low (it changed rapidly between sections of training data), and the major effort of subsequent iterations was to mitigate this problem.

The original Puppet Master algorithm included an explicit smoothing post-process where the output path was heavily smoothed to remove jitter. We recognized the importance of high-frequency movement texture, and as smoothing removed this texture (even when training was, e.g., purposefully shaky), we re-introduced the appropriate texture from the training data using wavelet-based frequency analysis: we extracted the high-frequency component from training and added it to our output. Unfortunately, as this was a post-process only the underlying search still maintained the jitter. Results were improved for minor jitter scenarios, but for more prolonged and pronounced jitter it still remained. The reason is that rapid movement between regions means that the training texture introduced to the filtered output itself may rapidly change.

For the robot follower implementation, the slow response time of the robot exacerbated the problem: any jitter at all made the robot move in a wrong direction, which is difficult for a slow robot to recover from in comparison to an animated sprite. Our solution here was to integrate the insights of the frequency analysis approach into the core algorithm instead of being a post-process. By splitting the output into the target location and detail (low and high frequency components), and balancing them, the robot was able to maintain relative-location coherency while still reproducing the texture as trained. Although the actual produced path was not the same as the original, the low and high frequency components were recognizable and the result was much stronger. Although we felt this solution was robust, our user studies indicted that people were very sensitive to any jitter and it remained a problem.

To reduce jitter for puppet dancer we realized that a solution must stop the jitter at the source instead of filtering it. We adapted the hysteresis solution (in addition to the frequency-balancing approach) to reduce jitter at the expense of some interactivity (the smoothing and wavelets technique was completely abandoned). We believe this was successful as no one in the puppet dancer user study complained of jitter.

Other than the above issues, the other main components of Puppet Master stayed the same, including the search (coherency and similarity balancing) algorithm, feature characterization, and so forth.

3.7. Summary of the Puppet Master SBD Algorithm

The Puppet Master algorithm is a pattern matching technique with patch-based output composition. The original contributions are in the adaption of a previous texture-synthesis technique to work with real-time feature data without available lookahead, and the frequency-analysis based approach to smoothing motion-path patch transitions in a way that maintains both movement texture and global position.

We note that our model of motion *style* relies on the high-frequency components of motion path. Further, our primary purpose of developing Puppet Master was not only to solve the hard SBD algorithmic problem itself, but to provide a learning system that sufficiently enabled flexible end-user authoring of style-oriented robotic behaviors for the purposes of evaluating the SBD interaction paradigm. Puppet Master is the first learning system to sufficiently meet these goals: it learns *interactive* behavior with a focus on *style* over *goal*, and it enables rapid and iterative behavior creation without lengthy expert-involved pre-processing.

4. OVERVIEW OF THE PUPPET MASTER SBD PROJECTS AND EVALUATIONS

In this section we briefly introduce our three Puppet Master incarnations and give a summary of their evaluations. The projects are presented in chronological order, to illustrate how the research evolved from early animation work (proof of concept) to a robot implementation and to adaptation to an entirely different feature set. We will follow with a cross project and study analysis where we investigate the overall SBD challenges and successes, reflect on how users engage SBD in general and how SBD interaction can be designed, and reflect on the challenge of evaluating SBD systems. As such, in this section we provide only enough summary information to clearly portray the interfaces, and to highlight the interaction and evaluation challenges and solutions explored; full details of each project can be found in their respective publications.

We conducted evaluations to test the usability of each particular interaction and interface design (i.e., if people will be able to use our interfaces to comfortably author interactive, stylistic behaviors), and to test the efficacy of the Puppet Master SBD algorithm, that is, if it produces satisfactory mimicry results that people can recognize and understand. However, our primary purpose in creating these systems was to serve as initial SBD proofs of concept where we investigate how people engage and interact with SBD systems, test if people easily understand the SBD paradigm of teaching interactive stylistic behaviors to robots by "acting," and confirm that people are in general motivated to engage SBD tasks. Thus, rather than focusing on quantitative measures such as learning accuracy or time efficiency, at this early point we believe it is more meaningful to take an exploratory approach to evaluation: we create believable SBD interaction scenarios which enable us to observe how participants perform tasks and qualitatively investigate how interaction takes place.

We rely heavily on qualitative evaluation methods, using participant self-reflection via, for example, verbal comments, long-answer questions and opinion-oriented Likert-like scales as data. Our specific method draws heavily from grounded theory: we open code the written answers, interviews, and video data, we perform multi-level analysis to identify emergent themes and relationships throughout the data, and we use exemplary quotations to represent the complex ideas [Bernard 2000; Strauss and Corbin 1998]. An interested reader can find many recent examples of this approach in the community (e. g., [Voida et al. 2011; Wahlström et al. 2011]).

4.1. The Animation Project

The animation SBD project enables users to create stylistic and interactive character locomotion: the way that one character moves in real-time tandem to the movements of another character [Young et al. 2008]. We note that an animated character can convey complex stories and personalities through its locomotion path only [Heider and Simmel 1944], for example, one character could be friendly or aggressive simply by how it moves around and in reaction to another character. We use two on-screen characters for the leader and reactor: during authoring users train a yellow slime how to interact with (via locomotion path) a green slime. During generation the yellow reactor moves automatically based on the personality demonstrated.

We constructed two interfaces, a mouse-based GUI and a Tangible User Interface (TUI)-based digital tabletop interface, to explore different interaction styles. The mouse-based GUI (Fig. 7) is a PC application for authoring interactive slime behavior. For authoring the user provides leader and reactor example movements sequentially. First, the user defines a leader movement path by clicking and dragging the cursor around the screen. When finished, the GUI re-plays the leader path verbatim while the user demonstrates reactor movements to interact with the leader. Finally, the user moves the leader and the reactor automatically interacts using the trained behavior.

A limitation is that the standard mouse does not have a rotation sensor, for example, the slimes cannot be backed up or moved sideways. We locked the look direction to the movement direction. Further, initial pilots revealed that users found the sequential nature of training to be unnatural, and they requested simultaneous authoring.

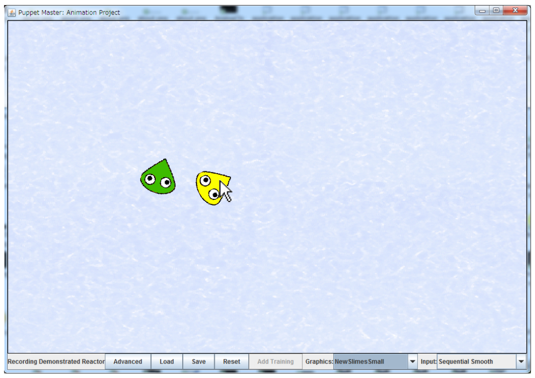
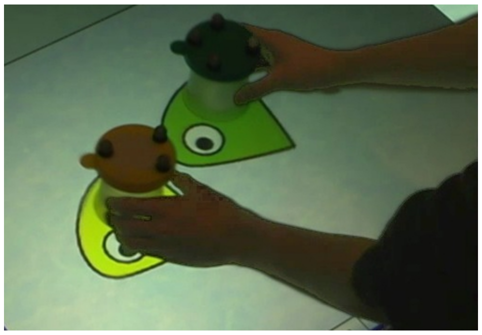
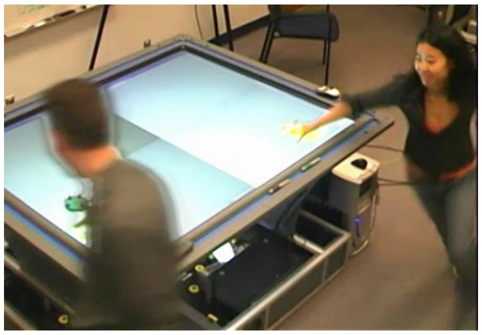


Fig. 7: The animation project GUI with a leader yellow slime and reactor green slime.

Our TUIs tabletop design (Fig. 8) allows people to use physical objects (the TUIs) to demonstrate behavior. This enables users to specify slime orientation, and the two TUIs enable a user to simultaneously manipulate the leader and reactor during authoring (Fig. 8(a)), or, two collaborating people to author together (Fig. 8(b)). This is reminiscent of puppetry, an idea that people are familiar with. Further, the TUIs improve usability by combining the action and perception spaces and by leveraging the physicality of the TUIs [Fjeld et al. 1986; Ishii and Ullmer 1997; Sharlin et al. 2004].


(a) Authoring how the green-slime reactor should interact with the yellow slime leader, controlling both simultaneously.


(b) Two users collaborating to *author* an interactive behavior.

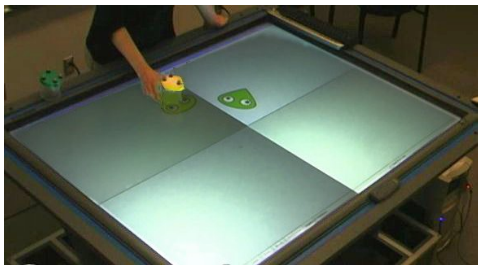

(c) For generation the green reactor interacts with the yellow leader in the style demonstrated.

Fig. 8: Using TUIs on a digital tabletop for SBD.

4.1.1. Animation Study. The animation study consisted of an authoring and an observing component, where one group of participants created and evaluated behaviors, and another group of simply evaluated them; we recruited 6 female/14 male, age 19–32, M=22.8. They were each paid $15 for 60 minute participation. We used written surveys pre-post test and throughout the studies as primary data.

Authoring participants designed character behaviors one at at time – the participant simultaneously demonstrated the leader and reactor paths – given the following keywords to describe the learning goal: *lover, bully, playful friend, stalker*, and *afraid*, chosen as a broad range of examples. Participants could re-try if they were not satisfied with the results, and questionnaires were administered after each behavior. Next, they were shown their behaviors in a random order and asked to match them.

Although we did not ground our behavior selection in behavior theory, and admit that this limits our ability to generalize our conclusions, the selection serves our primary exploratory purpose of creating scenarios that engage participants. It further provides a broad test of the robotic Puppet Master SBD capabilities: the Puppet Master algorithm has no hard-coded behavior model, and must learn each of these behaviors completely and wholly from a participant's demonstration.

Observer-study participants were first shown one of each behavior type from a set subjectively-selected by the experimenters from authoring-participants creations. After each they were asked to "describe the character" in a questionnaire. Following, participants were given the titles of the behaviors and asked to match them when shown again in a scrambled order. The study structure is summarized in Tab. III, page 25.

4.1.2. Results and Discussion. Observation indicated that participants were highly engaged with the SBD interface, task, and behavior results. Four authoring participants were observed to be highly immersed in the design process, for example, making exaggerated faces, noises, and speaking out loud to the characters while training. One participant hummed the "Jaws" movie theme while training the afraid behavior, and another commented "what a jerk!" when observing the designed "bully" character. No authoring participant exhibited problems understanding and executing the SBD tasks, and all 10 strongly agreed that they enjoyed using the system. Observer participants were observed to be role-playing their characters, and used anthropomorphism to describe their observations, for example "the guy who kept sucker-punching" and "he needs more confidence."

The authoring study results found that eight of the ten participants correctly identified all their own behaviors, participants were satisfied with the results 74% of the time, and that all particpants strongly agreed that they enjoyed using the system. Further, this was accomplished with no participant training, with on average of 32.5 s of demonstration, and on average 1.7 attempts at each behavior. Thus Puppet Master supports rapid end-user prototyping of interactive behaviors that captures the behavior style and personalities reasonably well.

Observer participants used exact or similar (e. g., "girlfriend" instead of "lover") keywords to describe the behaviors during the open ended phase in 38% of the cases. For the matching phase, behaviors were correctly identified 50% of the time (significantly better than random choice). The pattern of incorrect matches suggested crosstalk between behaviors, for example, *afraid* and *stalker* were often mistaken for each other.

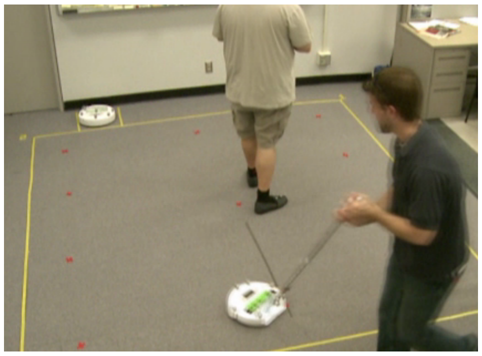
A common criticism across both phases was jitter in the reactor generation: people reported that it was distracting and made the behaviors feel fake. Another reported problem was the lack of character capabilities, for example, "he doesn't have hands so I can't punch," and several participants noted the lack of intelligence in learning, such as "if you pause to catch your breath, the system takes it as deliberate behavior."

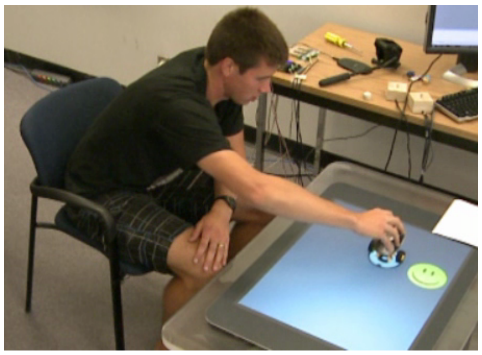
Overall, this study supports SBD behavior creation for locomotion paths, and shows that end-users will be able to easily understand the interaction paradigm.

4.2. The Robot Follower Project

This project enables people to design the interactive, stylistic locomotion of a robot reactor – an iRobot Roomba – interacting with a person leader [Young et al. 2010, 2012]. We created two authoring interfaces for this project: the *broomstick* interface, where a robot-on-a-broomstick is used to show interactive locomotion style (Fig. 9(a)), and the *Surface* interface, where a tangible puck on a tabletop Microsoft Surface computer is used (Fig. 9(b)). Both authoring interfaces are designed to physically constrain the demonstration movements which are reproducible by the robot: the interfaces, like the robot, can turn on the spot but cannot move sideways, forcing the person to express their desired movement style using the robot's movement capabilities and limitations.

For the broomstick interface, a person leader moves in a path and a second person manipulates the broomsticked robot reactor to shown how it should interact with the person (Fig. 9(a)). The broomstick design motivation was to enable free demonstration as directly to the robot as possible. Grasping the robot directly is not feasible due to the robot being low to the ground. The person cannot act using their own body as this would incorporate degrees of freedom not reproducible by the robot. The broomstick


(a) Authoring interactive robot locomotion style using the broomstick, showing a Roomba reactor how to interact with a person leader.


(b) Authoring using the Surface, with a TUI for the Roomba reactor and the happy-face representing the person leader example input.

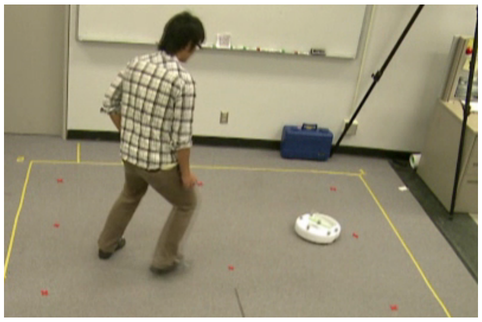

(c) For generation a Roomba interacts using the demonstrated behavior, enabling the person to experience the result directly.

Fig. 9: the *broomstick* and Surface SBD interfaces for robot locomotion path.

solution provides a fairly-direct demonstration method, while the robot on the end constrains demonstrator input to motions reproducible by that robot.

The Surface authoring interface (Fig. 9(b)) uses a digital tabletop (Microsoft Surface): the leader example movements are pre-scripted and presented as a happy face icon on the table, and the user manipulates a wheeled puck TUI (reactor) to show the robot how to interact with the person's movements. We pre-scripted the movements for simplicity of implementation, and as the human movements for the broomstick were also rigidly scripted we believe it was not necessary to have these controlled live, and having a person perform on the same small tabletop introduces occlusion issues. This design was an explicit follow-up on the animation project and the results of the related study: the Surface is smaller than the previous table, such that the demonstrator can easily reach over the entire space while maintaining smooth motions, and we avoided having two pucks to lower mental load [Young et al. 2008]. The puck was a hand-made wheeled platform with a wireless mouse attached.

For generation the leader person simply walks around the space freely while the reactor robot interacts using the demonstrated style (Fig. 9(c)). The broomstick robot and leader person's movements are tracked using a Vicon camera motion-tracking system.

4.2.1. Robot Follower Study. The robot follower study consisted of a programmer design critique, and authoring and observing phases. In all cases we used four stylized interactive robot locomotion behaviors: a polite follow (*polite*), stalking the person (*stalker*), happy to see the person (*happy*), and attacking the person as if they were a burglar (*burglar*); the behaviors used in the animation study were adapted here to the task of "following" and more words were added to improve clarity (e.g., compare the *burger* description above to simply "*bully*"). Further, in all cases an experimenter performed leader movement paths for the reactor to interact with. Movement paths varied greatly by study phase (but were consistent for behaviors across participants), and for generation the participant watched the reactor robot interacting with the leader actor.

For the programmer design critique we recruited 4 experienced programmers (1 female / 3 male) and asked them to create interactive robot behaviors both programmatically (via our easy-to-use Java API) and using our SBD broomstick interface with an actor as the leader. Participants were given two hours to program four different behaviors, and no time limit to create and evaluate the same behaviors by SBD. We video-taped the study and interviewed participants post test.

Twenty-two participants were recruited for the authoring phase and were split evenly between the broomstick and Surface interfaces: they were 11 female / 11 male, aged 19–34 (M=26.9), and were paid $20 CAD for one-hour participation. Participants were first asked to create the four behaviors in order, using the assigned SBD interface, observing the result and completing a questionnaire after each. They could re-try a behavior until they were satisfied. Following, participants were asked to identify their behaviors when presented in a shuffled order.

For the observer phase we recruited 12 additional participants (5 female / 7 male, aged 19–36, M=26.3) for a one hour study. Participants first simply observed a robot reactor with an actor leader for each behavior type, then were informed of the behavior categories and asked to classify a new set (which included the programmer behaviors) while watching again, and were finally given the option of an unstructured task where they become the leader and interact with the robot in-situ; this last phase was simply an opportunity to observe organic interactions with the stylistic robotic behaviors.

Both the authoring and observer phases were video taped, and free form and Likert-scale questionnaires were administered throughout the studies. The study structure is summarized in Tab. III, page 25.

4.2.2. Programmer Results and Discussion. The programmers embraced the SBD approach: they anthropomorphized the robot and “played along” with the behaviors, despite their technical view. The primary finding was that, rather than categorizing SBD or programming as one being better than the other, all four participants described an accuracy and control versus time and ease trade-off: participants reported that programming gives explicit control but demands considerable time, and it is difficult to convert stylistic ideas into raw movement commands. On the other hand, SBD gives a direct, quick, and easy method for showing intention at the cost of losing detailed control of the robot’s movements. These trade-offs are not clear-cut, as participants pointed out that programming gives a false sense of control: “even when I program I don’t know exactly what is going to happen.” Programming tools are also perhaps ill-suited to the task: “when you’re programming something you have to anticipate ... what kind of situations can come up and how [the robot] should react ... that’s not a natural way of doing things.” Further, it was pointed out that SBD cannot be a perfect solution because the algorithm is “relying on its interpretations of [their] intentions, rather than on [their] actual intentions. There is no way to directly convey intentions.”

All participants took the full two hours to program the four behaviors, and took an average of 10 m 27 s total in the SBD task, including iterative demonstration, observation, and simple discussion; the SBD approach obviously saved time, a benefit which would be dramatically stronger for non-programmer users.

Overall, this study suggests that the SBD approach may be useful even for users who have the capability to program behaviors more explicitly; it still makes sense to leverage people’s demonstration abilities. Further, it highlights important limitations and benefits in comparison with traditional methods.

4.2.3. Authoring and Observing Results and Discussion. The authoring study results show that people understand and accept the SBD approach: no participant was observed having problems understanding the SBD concepts, tasks or interfaces, and many applauded the idea of easy customization, for example, “each person’s interpretation of aggressive would be different, it wouldn’t make sense to pre-program.” The Puppet Master algorithm enabled people to quickly (M=50 s and M=1.27 tries) create interactive, stylistic robot locomotion behaviors that they were satisfied with and could reasonably identify (67% overall): half the participants perfectly identified their behaviors. 67% was less than hoped for, highlighting the need for improvement. For the observer study overall match success was 54%, which was much better than chance alone (t(11)=6.13, p <.001). Further, there was a clustering of similar behaviors, that is, polite and stalker, and burglar and happy were often mistaken for each other.

As in the animation study, some felt restricted by limited robot capabilities, for example, “a dog would run around, jump, move its tail and follow its owner.” The most frequent complaint was that the robot’s movements “seemed really jerky.” However, unlike in the animation study, this was in the majority of cases interpreted as a personality trait: that the “robot seemed a bit confused” (the most common attribution by far) or that it “looked like [sic] thinking and deciding.” Further, the same relationship emerged in the observer study where there was no context of teaching and learning which may create such an interpretation.

Eleven of the 12 participants opted to participate in the in-situ free-form phase. One result was that the level of anthropomorphism and “buy in” for the social robotic personalities was informally observed to increase for participants in comparison to the less direct behavior observation phase.

Overall, the authoring and observer study, as in the animation case, provides support for the approachability of the SBD technique to untrained users.

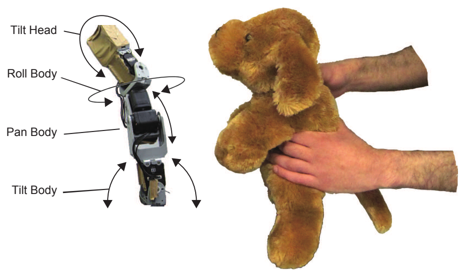
4.3. The Puppet Dancer Project

The puppet dancer project [Allen et al. 2012] is an SBD system for enabling people to create interactive robot dances (Fig. 1, Fig. 10). We selected dance as: a) dance is heavily style oriented and generally lacks an overarching task which must be considered, and b) paired dance is highly interactive as we envision many robotic tasks will be.

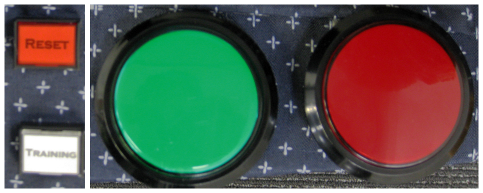
This interface enables a user to teach a reactor dog robot how to interactively dance with a cat leader by providing a demonstration (Fig. 1). This is a paired dance where the reactor responds directly to the leader's motions, for example, if the leader dips the reactor may dance from side to side. After authoring, during generation the reactor dog will automatically dance in the fashion demonstrated as the user manipulates the leader cat (Fig. 10(a)). The robots are stuffed toys with robotic spines (Robotis Bioloid servos, Fig. 10(b)) that enable both robots to 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 (Fig. 10(b)). Further, both the cat and dog are mounted rigidly facing each other on a plastic box containing the controller boards (Fig. 1, Fig. 10(a)).

The robot dancer SBD interaction design was modeled after how people interact when teaching one another. First, the user can switch seamlessly between authoring and generation, to alternately provide additional training and test the incremental result. This is modeled after how human teachers will regularly stop a progressing human learner to provide additional input, for example, saying "like this" while giving a physical demonstration. This is enabled via a toggle button with the label "training" on it (Fig. 10(c), left): when depressed, it blinks and robot dancer enters authoring mode, when toggled out, the light extinguishes and robot dancer enters generation mode. In addition, there is "reset" button which, when pressed, makes the reactor forget all training (Fig. 10(c)). This design was a deliberate attempt to make training more free-form than prior Puppet Master systems; we analyze this directly in Sec. 5.

(a) Dance generation: the reactor dog automatically dances along with the leader cat.



(b) The reactor dog and embedded servo spine.



(c) The buttons used for robot dancer control.

Fig. 10: The Puppet Dancer interface.

Additionally, we enabled users to give real-time feedback to the reactor regarding if it is doing well or not, interaction modeled after how teachers will often give real-time verbal feedback to guide a student working on a new task, for example, saying “yes” or “no” as they try. For this we mounted two arcade-style press buttons on the top of the robot platform (Fig. 10(c)). The user can press the green *approve* and red *disapprove* buttons to signal to the reactor when they like or dislike the dancing.

4.3.1. Puppet Dancer Study. The puppet dancer study consisted of a creation / comparison phase to compare Puppet Master against other dance generation methods (remote control and non-interactive), a feedback phase to target the reinforcement learning mechanism, and an open-ended phase to explore how participants engage the overall training process when not given constraints; the overall structure is summarized in Tab. III, page 25. We recruited 11 participants for one hour participation; 4 female / 7 male, aged 20-52 (M=29.0). Some analysis numbers total less than 11 due to early study finishes from robot and camera problems. We administered questionnaires pre- and post-test, and after each phase.

In the comparison phase participants freely authored an interactive robot dance – for as long as they liked – by manipulating the reactor dog while an experimenter moved the leader cat (pre-scripted). Following, the participant controlled the leader while the reactor interactively danced along, to evaluate the three reactor cases in turn (Puppet Master, remote control, non-interactive, presented in a counterbalanced order); they were not informed of the differences and evaluated each one immediately after interacting with it. Remote-control was achieved via a hidden web-cam and remote-control with the same morphology as the dancing robots, and random replay was achieved by selecting random patches of training data instead of using the algorithm's “best match,” and still using Puppet Master's output mechanism which smoothly transitions between patches through its emphasis on movement texture by sacrificing localization: the result was a smooth dance taken from the training but was not interactive. Here, the training, reset, and feedback buttons were all hidden.

Next, participants were introduced to the feedback (*approve*/*disapprove*) buttons. They manipulated the leader while simultaneously observing the reactor and using the reinforcement buttons to shape the result. We used Puppet Master generation with training data from the first phase. There were two parts, the standard reinforcement method and a variant where button presses has no effect. We believe this comparison was valid (instead of, e. g., comparing to random effect) as, even with the buttons enabled, the effect was not immediately obvious due to the small weights.

In the unstructured open-ended phase participants were shown the buttons that enabled them to add training incrementally or to “reset” learning. Participants were encouraged to simply play with the system as they wish for as long as they wish. For authoring they themselves manipulated the leader and reactor simultaneously.

4.3.2. Results and Discussion. Participants were engaged, clearly understood the tasks with minimal instruction, and were able to author a wide range of interactive dances. Overall, they reacted positively to SBD, and there were only a few negative comments, primarily about the learning ability and result. No participants complained of jitter.

For the comparison phase, there was an effect of generation type on how participants ranked the quality of the generation (Friedman's ANOVA on Likert-like scale responses, $X^2(2) = 4.923, p < 0.05$); post-hoc tests failed to reveal additional effects, Friedman rankings: Puppet Master 2.0, wizard 2.35, non-interactive 1.64. Participant feedback on questionnaires had a similar pattern, with Puppet Master and remote ranked similarly with non-interactive lagging behind. Thus this supports the importance of having interactive stylistic dance instead of static pre-scripted approaches.

Most participants reported that they found the reinforcement-learning buttons useful, and several related the importance to the feeling of teaching: "having the like and dislike buttons I really felt that the robot is learning." Although participants rated the enabled reinforcement-learning condition as being more effective than the (unknown to them) disabled condition ($T = 2.5, p < 0.05, r = -0.47$), we found no effect of this condition on how participants actually used the buttons (e. g., frequency, which ranged from 3–49 hits across participants). When asked overall whether the buttons were useful, 3 disagreed, 1 was neutral, and 5 agreed, participants cited heavily in written answers the importance of the reinforcement mechanism, and were observed to use the mechanism extensively in the open ended phase. This highlights the importance of providing SBD users with a fine-detailed and incremental way of modifying their behavior without resorting to the core demonstration mechanism.

Our open-ended phase analysis found that individuals engaged SBD and used the tools in a surprising breadth of ways. Some built behaviors piece-wise, others created them at once monolithically, and others relied heavily on reinforcement to shape the result. No "average" user emerged from our study and there was no overarching theme or method to dance creation or engaging SBD. Many participants highlighted in written feedback the importance of the various components to support exploration. This points to the importance of having a flexible SBD interface that supports various rapid prototyping and refinement methods to match individual preferences and styles.

We analyzed our video data to investigate if participants would simply want a mimicking reactor or a more freeform dance. 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. Another finding of note was that nearly half participants (across the cases) were found taking a deliberate *leading* behavior, where they would, for example, do a movement with the leader and then stop to observe the reactor, testing for things that they taught. Thus this supports the idea that participants will want individualized and personal dances and not mimicry only.

The results of the puppet dancer study further lend support for the success of the SBD approach, for behavior types other that interactive locomotion paths. Further, the open-ended interaction nature of this study helped highlight the different ways in which participants may engage an SBD interface.

4.4. Overall Results

Above we detailed three different SBD interface design approaches and realizations, and a summary of several user studies. The results support our primary research questions, that is: a) people understand the "acting" concept of SBD and can quickly use it to author custom interactive robot behaviors with minimal to no training, b) our interfaces support various authoring and interaction styles to enable SBD, and c) the underlying Puppet Master algorithm is reasonably successful in learning the interactive stylistic behaviors and generating convincing results.

Although we did not conduct benchmarking of the Puppet Master algorithm, in all the studies computational limitations were not reached; informal testing found that saturation happened at about 10 minutes of training data, where output began to stutter. The current algorithm was not optimized, and future work could improve speed, for example, by using caching techniques, parallelizing the search, or approximate optimizations such as approximate nearest neighbor.

Directly comparing quality of generation between the interface instances is problematic given the differences in the behaviors and interfaces, as well as stark differences in how each one was evaluated. However, informally we note that generation quality – and user satisfaction – improved through the iterations. We believe this was due to improvements in the base algorithm (primarily removing jitter) and having a more free-form interaction setup; this is addressed more in the next section.

More importantly for this paper, the above discussion detailed the various SBD interface designs which we explored, and the various evaluation methods applied. Below we take a closer look at how these decisions varied across the studies.

5. CROSS-PROJECT AND CROSS-EVALUATION ANALYSIS

In this section we present a macro analysis of our three SBD platforms and evaluations to investigate SBD in a broader way than is possible with any single project, to reflect on how our interface design decisions impact user experience, and to reflect on the various evaluation design choices tested. Overall, we expect that our analysis will reflect fundamentally on SBD above and beyond our specific platforms, and will provide insight into how future HRI interaction techniques can be designed.

We split this presentation into two sections: first we discuss the SBD interface designs, and then discuss our approach to SBD evaluation.

5.1. Analysis of SBD Interaction Design

A primary SBD interaction design challenge that we explored through our projects is how to best scaffold authoring to help a user define their desired behavior. We explored such questions as:

— should a user perform leader and reactor authoring movements simultaneously or sequentially
— should we have two people, one for the leader and one for the reactor, or should a single person control both
— should we structure the authoring process or keep it more free form

With this in mind, the following discussion focuses on two key components of SBD interaction design that we varied between projects: the user's role in interaction, and the authoring and generation interaction process.

5.1.1. User's Role in Interaction. We explored the question of whether a single person should control both the leader and reactor simultaneously for authoring, or whether two people should demonstrate in tandem; Tab. II provides a breakdown of the roles assigned throughout the studies.

In the animation project we took the solo single-user approach in order to give the user full control; although the tabletop interface supports two users in tandem, our studies did not take advantage of this. Study results (and pilots for later studies) suggested that participants had difficulty concentrating on both the leader and reactor simultaneously for authoring. One alternative we explored with the animation mouse GUI interface is sequential demonstration, where the user first demonstrates the leader movement, and following, the leader is replayed and the user demonstrates the reactor's response. Pilot testing showed this to be highly confusing and difficult to use, as people wanted to control the leader and reactor in real time.

Following from this problem, for robot follower we decided to employ a second person (an experimenter in our studies) to perform some of the leader roles, as a way to en-

Table II: Who performed the leader and reactor roles for each phase, "pct." is shorthand for participant.

	authoring		*generation*	
	leader	reactor	leader	reactor
animation	pct.	pct.	pct.	generated
robot follower	actor	pct.	actor	generated
puppet dancer	actor / pct.	pct.	pct.	generated

able participants to focus on the reactor training only. While this hinders user demonstration freedom as they cannot control the leader during authoring, in our study no participants complained or commented on this limitation.

As an attempt to lower these restrictions on author freedom while still considering user mental load, for puppet dancer we took a hybrid approach. Initially in our study, to keep participant cognitive load low, the leader was controlled by an actor. However, after participants gained experience we had them control both the leader and reactor simultaneously for authoring. This had the desired effect of easing users into the SBD paradigm and enabled them to still have freedom later in the study. Although people complained about the lack of freedom during the early phase that involved an actor, unlike in the animation study no participants complained of the mental load required to simultaneously author the leader and reactor.

The task difference between the animation and puppet dancer cases makes it difficult to make strong conclusions, but at the least these results suggest that a) simple training may mitigate interface complexity problems, and b) single-user simultaneous training of both the leader and reactor is feasible and desirable for some SBD tasks. An open question here is the problem of how to best design an interface to support single-user paired authoring or multi-user tandem authoring, for example, how could a single user demonstrate both roles in the robot follower case?

5.1.2. Authoring and Generation Interaction Process. We designed the authoring and generation interaction process in the early Puppet Master projects (animation and robot follower) for simplicity: users could demonstrate a behavior, see the results, and start over if they wish. The studies showed that, as designed, this minimized training required, and participants did not complain of limitations or request more functionality.

With puppet dancer we explored additional functionality (incrementally add training, reinforcement) to investigate if people would use the functionality and, if so, how they would use it. The study results showed that not only did users find both the reinforcement and “add training” features to be very useful, but also that they used the various features in completely different ways to build a behavior. Although we did not compare the quality of the authored behaviors between the earlier limited version and the one with more functionality, these results suggest that SBD interfaces can benefit from providing a flexible range of behavior generation and fine-tuning mechanisms; users benefited from having the freedom to explore the authoring task in various ways. Further, the success of the more-limited versions suggests that, although extended functionality may be beneficial, it is not required for SBD to be successful.

As mentioned early in the paper, one limitation of our interaction process is that our leader-reactor role setup is not inherently reciprocal, where the leader is expected to simply move freely and the learner should adapt. In our studies, we have both a rigid leader (e. g., pre-scripted and controlled by an experimenter) or flexible one (controlled by the user). We did notice that users would modify the leader movements to try and elicit particular responses (both informally in the animation study and through video analysis in the puppet dancer study); it will be interesting for future work to consider how to integrate such behavior into the learning itself, for example, by identifying modified leader behaviors and tuning the reactor learning accordingly.

5.2. Analysis of SBD Evaluation Design

One problem we faced for our study design is a lack of similar SBD systems against which we could compare. While there are many PBD implementations available, they either a) do not create interactive behaviors (results are static), or b) do not focus on style but rather on task-level goals, or c) require detailed expert pre-processing and very large training databases which would prohibit our rapid and iterative behavior design paradigm. Likewise, there is a lack of SBD evaluation methods which we could directly use in our work, and the broader problem of how to evaluate the social aspects of HRI itself is yet not solved (see [Fernaeus et al. 2009; Young et al. 2010]).

Another problem is a lack of metrics for evaluation and comparison. Similar work developed metrics, for example, for comparing how "guessable" or "similar" pen-based gestures are to a trained example [Long et al. 2001; Wobbrock et al. 2005]. Our problem is much more difficult, stemming from the interactive two-entity ("paired") demonstration and that the temporal order of demonstrated features has little bearing on the result. For example, while demonstrating a similar behavior one person may provide a long demonstration repeating one important aspect several times and then give a single example of other aspects, while another person may simply give a quick but clear demonstration with one example of each feature. Numerically comparing these two examples is quite difficult. Other similar prior work has people rate how natural a result is, for example, for learned robot interactive eye gaze patterns [Mohammad et al. 2010]: in our flavor of SBD the goal is to enable a user to create their own desired style and not something which would be natural to most people (at least, within a cultural setting), and thus such generalized metrics do not quite apply here. We leave the development of comparison metrics as important future work, and in our studies, focus rather on qualitative results of how users engage and experience our SBD platforms.

Due to the lack of methods for SBD evaluation and lack of related work to compare to, through our studies we explored various evaluation techniques for investigating SBD research questions. In this section we summarize some of our methods and discuss the advantages and limitations of each. For reference and as an aide in discussion, Tab. III provides a breakdown of the various studies and components.

5.2.1. Exploratory Qualitative Approach. Across the studies we used exploratory qualitative evaluation techniques as a means of developing understanding of how users will engage and use SBD systems. Our data collection methods include think-aloud exercises, open-ended questionnaires, interviews, and video recording, all of which yielded important insights into interaction. Analysis techniques used were primarily data-driven approaches heavily inspired by grounded theory [Strauss and Corbin 1998] and included open-coding and axial coding, affinity diagramming, and multi-level code analysis. This approach has been successful, as the studies improved our basic understanding of how people engage SBD as well as specifics such as the varying ways in which people interact (i. e., there is no clear "average" user).

A drawback has been that we lack concrete criteria and metrics for evaluating our SBD interfaces and interaction designs. This makes it difficult to compare our results to future and similar work in the area.

5.2.2. Prescribed Behaviors. A primary evaluation design goal across the projects was to get people engaged in a task so that we can perform our exploratory qualitative evaluation. Although initially we tried to keep experiments as free-form as possible, participants in early pilot studies expressed difficulty being creative given the limited ecological validity of the study. Thus, we prescribed behaviors for participants to create in the animation and follower projects. This also served the purpose of standardizing behaviors across participants and providing a means of between-participant analysis.

We found that comparing behaviors was difficult as people often have dramatically different interpretations of high-level behaviors such as *angry* or *happy*. We could mitigate this by further restricting behavior freedom and clearly defining concrete behavior descriptions (perhaps with a video), effectively trading creativity in favor of easier analysis. However, this would have to match the evaluation purposes: for us, freedom is more important as we explore SBD interaction design and not behavior results.

Table III: A breakdown of the three evaluations discussed in this paper, with participant counts, and how the *authoring*, *observation*, and *programming* phases relate. *Pct.* is shorthand for *participant*.

	study	ptcs.	task	procedure	purpose
animation	auth.	10	behavior creation	create: *lover*, *bully*, *playful friend*, *stalker*, and *afraid*	engage pct. in SBD for a range of behaviors
			behavior match	interact with just-created behaviors in a shuffled order and identify	engage pct. in SBD for a range of behaviors test Puppet Master's learning
	obs.	10	open-ended observation	pcts. who did not do authoring, watch behaviors and think-aloud	elicit open-ended reactions to the behaviors before biasing pcts.
			behavior matching	match behaviors to given titles	test if informed pcts. can identify behaviors they did not create
robot follower	prog.	4	programing	create: *happy*, *polite*, *burglar*, and *stalker* by programming	give programmers experience creating and thinking of behavior to serve as comparison baseline
			SBD	create: *happy*, *polite*, *burglar*, and *stalker* with broomstick interface	explore expert programmer's experiences with SBD, compare to the programming approach
	auth.	22	behavior creation	create: *happy*, *polite*, *burglar*, and *stalker* using broomstick or Surface	engage pct. in SBD for a range of behaviors
			behavior match	observe actor interacting with just-created behaviors in a shuffled order, and identify	engage pct. in SBD for a range of behaviors test Puppet Master's learning
	obs.	12	open-ended observation	pcts. who did not do authoring, watch behaviors and think-aloud	elicit pct. reactions to compare against pcts. who trained
			behavior matching	match behaviors to given titles	test if biased pct. can identify behaviors they did not create
			in-situ evaluation	pct. open-ended interaction with robot	test pct. interest compare pct. reactions when observing versus interacting
puppet dancer	auth.	11	behavior creation	create a single robotic dance with zero training, only one try	engage pct. in SBD test acceptance of interaction and interface with minimal experience
			behavior comparison	interact with behavior created by Puppet Master, remote control, or non-interactive	compare Puppet Master reactive generation against remote-control person and non-reactive replay
			reinforcement learning	interact with reinforcement interface with buttons having / have no effect	test pct. reactions to giving reinforcement for SBD
			free-form	unstructured interaction with full platform	explore how pcts. will engage SBD, use behavior resetting or additions, or reinforcement learning, when not given structure

With the puppet dancer pilots we noted that, in comparison to the locomotion path work, participants showed no problem being creative. Thus we took a free-form approach and did not attempt to compare behaviors across participants. Perhaps creativity was easier here because of the dancing task (in comparison to locomotion path).

5.2.3. Experimenter Actor Involvement. For both the robot follower and puppet dancer studies we employed an experimenter ("actor") in authoring, to control the leader while the participant manipulated the reactor in response; actor involvement is summarized in Tab. II, page 22. This simplified interaction and enabled participants to focus on the authoring task, and kept movement consistency throughout the study to aid in cross-participant comparison. However, similar to using prescribed behaviors as discussed above, this imposed restrictions on authoring and limited creativity. If we intend SBD platforms to enable freedom of expression and authoring, then moving forward we need to consider how limiting participants in this way impacts the study results.

With the animation study and for a later puppet dancer phase, instead of employing an actor, study participants authored the leader and reactor simultaneously. This suited our exploratory purposes to examine how people engage the interaction task when not being led. Further, although for animation participants noted the challenge of controlling two entities simultaneously, this was not raised with puppet dancer, perhaps due to the “easing-in” of interaction with an earlier phase using an actor.

As an alternative to actor involvement we explored two participants working together (one for each the leader and reactor) as a means of lowering interaction complexity. We avoided this in our work as pilots showed that it created complications in evaluating the quality of results, for example, the leader participant evaluating results without knowing the reactor participant’s intentions. However, the question of how two people may interact together is an important one for future work.

In the robot follower study only, we employed an experimenter actor during generation to play as the leader, so that the participant could watch the generated reactor results without having to focus on their own interaction; we selected this design in part as pilots showed that people were constantly watching over their shoulder and stopping and observing, and did not tend to walk naturally as was needed. Informally, we did find differences between people observing robot interaction from the sidelines and interacting themselves (e. g., level of anthropomorphism). Thus future work should explore how the directness of interaction relates to evaluation goals and results.

5.2.4. Real Robots Impact Results. Although movement jitter was a problem for the animation and robot follower projects (the algorithm was improved for puppet dancer), there was a strong difference in *how* people interpreted the jitter. For the robot, it was attributed to the robot’s personality and seen as a sign of confusion, while for animation, it was seen as simply a bothersome software error. This suggests that the embodiment of the movement impacts how generation is received, a finding which agrees with a plethora of work which highlights how robots encourage anthropomorphism (e. g., [Sung et al. 2007; Young et al. 2009]). Thus, for investigating interaction with robotic SBD systems it is important to use real robots instead of on-screen simulations.

5.2.5. Involving Programmers. Ultimately, SBD will be but a part of a larger behavior design system. By involving programmers to compare SBD to programming, we identified ways in which SBD can be integrated within a larger framework. Programmers have a unique perspective in that they can appreciate the benefits of SBD but understand the technical context within which the results must work.

5.2.6. Evaluating SBD Generation Results. We explored various methods for evaluating the performance of the Puppet Master SBD algorithm generation. This evaluation is a non-trivial problem as there is no SBD standard to compare to, and no prior SBD evaluation work to build on. Further, SBD generation is subjective by definition, making it difficult to define generation performance metrics; not only should people be happy with their own creations (internal validity), but we must consider how results are perceived by others (external validity).

For the animation and robot follower projects we tested if authors could identify their creations. This method successfully tested the internal validity of the algorithm, that is, if Puppet Master captures what authors expect for their behavior. We further solicited participant satisfaction ratings of the generation quality, although we note that without a comparison point it is difficult to make conclusions based on this data.

We recruited people to evaluate behaviors authored by prior participants, to test if appropriate features were sufficiently captured to convey core behavior characteristics to others (external validity). We first did this in an open-ended fashion, where participants were asked to interact with the behaviors and to *think aloud* about their reactions. While we markedly avoided language that would suggest anthropomorphic behaviors, in retrospect this was a mistake: some found the task of explaining interactive motion path confusing without more context, and some thought they should explain technically, using angles and speeds. While overall this approach helped us understand how people interpret locomotion paths, the lack of context and explanation made it difficult to use it to reflect on Puppet Master generation quality.

Following this we asked participants to match shown behaviors to a set of given ones. This added context and so was more successful at testing whether people perceived the behavior similar to the authors' intentions. However, there was still a problem of ambiguity: for example, *stalker* and *afraid* reactors were often mistaken for one another, as they both often tried to stay out of the leader's sight. In retrospect, we believe that the approach of testing identifiability in an abstract setting — without sufficient scenario context – is flawed, as SBD in use would be strongly embedded in a task. If one is to take this approach, we recommend taking care to build context.

Instead of testing behavior identifiability, for puppet dancer we compared the Puppet Master results against other generation methods: remote controlled human operator and non-interactive random replay of demonstration. This provided a generation comparison where no other SBD system exists, and enabled participants to select which method they prefer to create their results. In this case we saw the human as a best-case comparison point, although our non-expert-actor experimenter found the open-ended mimicry task to be very difficult. We found this method to be successful, although for future work we would like to compare to other automatic generation methods.

Overall throughout the projects, we fundamentally had difficulty determining when the algorithm failed, when people were simply doing a poor job of demonstrating what they wanted, or if the limited robot capabilities hindered representation of the desired behavior. This could be mitigated by, for example, improving participant training or giving strongly-scripted behaviors (that match the robot) to create instead of keeping it free-form. While the latter hinders creativity, this can be acceptable if the evaluation goal is to test the algorithm and not the interaction.

5.3. Analysis Summary

Here we summarize our SBD interaction design and evaluation approach analysis.

5.3.1. SBD Interaction Design

USER ROLE IN SBD INTERACTION. Two people can work in tandem to demonstrate leader and rector movements, although this hinders author freedom as they can only control one character at a time. One person can effectively demonstrate simultaneous leader and reactor motions for authoring, but for some applications controlling two entities at once can be mentally challenging. There is some evidence that this can be mitigated by simple training.

AUTHORING METHOD. Provide various means of authoring and shaping a behavior to support differences in peoples' teaching styles. However, simply enabling rapid demonstration-test-re-demonstrate prototyping is sufficient to enable people to explore and create behaviors they are happy with.

5.3.2. SBD Evaluation Design

QUALITATIVE METHOD. Qualitative evaluation is an effective means for investigating interaction experience design goals (how people engage and use an SBD interface) when quantitative measures such as task completion time are less relevant for the given task.

PRESCRIBING BEHAVIORS. Prescribing behavior types can be an effective way to have participants create similar results which can be compared. Provide a high level of detail to avoid confusion and maximize comparability. However, this limits creativity, and when exploring interaction experience and interface engagement strategies it may be better to allow free-form interaction instead.

ACTOR INVOLVEMENT. Employing an actor to aid in tandem demonstration can improve study consistency across participants, lower required participant mental load, and simplify demonstration when controlling two entities is difficult (e. g., as with robot follower). However, having an actor restricts participant demonstration freedom and inhibits creativity.

USE REAL ROBOTS. Robots encourage anthropomorphism and thus fundamentally impact how the SBD generation is perceived. Avoid on-screen animation simulations and use real robots when possible to maximize validity of results.

INVOLVE PROGRAMMERS. Experienced programmers can still benefit from SBD: including them in evaluation design can provide insight into how the SBD system may be integrated into a platform.

GENERATION QUALITY. Metrics need to be developed for assessing SBD generation quality; in summary, we found testing internal validity (user satisfaction) and external validity (recognition by others) to be promising directions. We found author self-assessment to be effective for testing internal validity, but testing against a similar system (or variants of itself, as with Puppet Dancer) can provide more generalizable results. When testing external validity with non-authors be careful to provide sufficient context to creating ecological validity for fair evaluation.

6. FUTURE WORK

Here we present an overall discussion on open questions and directions for future work for interaction design, SBD algorithms, and evaluation approaches.

6.1. Evaluation Methods

As highlighted through our cross-project analysis, we must develop guidelines and metrics for evaluating and comparing SBD systems. Part of the problem of evaluation is that there are many competing angles of evaluation, with all being important for a particular system. In our work we identified that it will be important to:

— evaluate interfaces and interaction processes which support creativity (usability)
— evaluate the SBD generation results (the algorithm)
— evaluate if a user is a *good* teacher or a particular demonstration was a *good* one for the learning purposes (input validity)

In our work, separating these components was not always trivial as new algorithms, interfaces, and interaction designs were all created in tandem, but we hope that we can develop these evaluation directions more fully for future work.

A recurring theme in our own evaluations was the idea of internal and external validity of a demonstration: approval by the demonstrator (internal) and others (external). For future work we hope to standardize our measurement techniques for these such that results can be compared across systems more readily. Similarly, we intend to follow similar techniques as used in Puppet Dancer where generation results were compared against random variants and a wizard, to provide a means to compare projects, as well as to further investigate this evaluation method.

6.2. Beyond Paired Interaction

The current Puppet Master algorithm and interfaces are designed for paired leader-reactor behaviors only. It should be extended to enable more complex and diverse scenarios with multiple people, robots, and consideration of more detail such as people's body language. While there is a plethora of work which learns crowd behavior from examples (e. g., [Lerner et al. 2007]), these primarily focus on collision avoidance or macro-level crowd movement. The interface design of having multiple entities will face many of the challenges already discussed: should a demonstrator operate all simultaneously or in sequence, should other people be employed, and so forth. Algorithmically, it may be useful to consider how such groups could be represented as a statistical distribution, for example, representing a group as a mean (center) and standard deviation (measure of size and dispersion).

6.3. Evolving Behaviors

Similarly, the current current systems only enable creation of simple behaviors that do not evolve or change over time; this is why the behavior can be created with just a few minutes of training data. Future work should investigate how to author higher-level behavior changes such as a happy robot slowly calming down. The interface design challenge of enabling people to demonstrate such changes is non-trivial, and will require a re-consideration of the approach. Algorithmically, one way of implementing more complex behaviors would be to create a set of Puppet Master behaviors, and then develop a higher-level behavior manager that monitors interaction and somehow decides (e.g., based on high-level features such as character fatigue) which Puppet Master behavior should be employed at a given time, or even somehow combines them. Transitions between behaviors are expected to be smooth as a strength of Puppet Master is how it can maintain appropriate movement texture in the face of unexpected (or suddenly-changing) trajectory targets.

6.4. Interaction Process

The culmination of our interaction process development was in the puppet dancer project, which included mechanisms for real-time feedback and behavior shaping: adding additional training and using the reinforcement buttons. However, in real interaction, feedback from a teacher is more comprehensive than the good / bad mechanism we provide, more natural than the mode-switching training / generation technique, and often reciprocal in how the roles play out. Future systems should investigate how to integrate and enable more natural real-time feedback. One example is to automatically recognize when a user is trying to give feedback or a new demonstration, thus removing the need to mode-switch between training and generation. Or, perhaps if feedback contradicts a recent movement this could be detected and taken as re-enforcement learning instead of requiring the user to explicitly hit the "bad" button.

Algorithmically speaking, Puppet Master's current reinforcement learning and integration of additional training is rudimentary only, enabled for the purposes of testing the basic interaction concepts. Additional work in this area should address these areas more appropriately and investigate advanced techniques such as analyzing existing training and replacing components with updated training (instead of just adding more training as is currently done).

6.5. Context of Interaction

One of the goals of the Puppet Master interaction designs has been to simplify interaction to enable people to focus on a general behavior they want to create. A downside of this is that the approaches remove much of the context which would exist and be important in real-world interactions, for example, a dancing robot would react to a particular song, or a following robot would react to obstacles, other robots, and the social situation. The current interface designs do not provide any mechanism for including context or, if it was there, indicating which components of the context are important. A simple approach could be to include key elements in the interaction space (e.g., obstacles, or enabling the user to change the song for dancing robot).

Algorithmically, Puppet Master could include additional context elements as additional features in its best match searching, although there remains a question of balance between how context and current-behavior features should impact the result. For example, should context features be weighted heavily to trigger modal-like reactions but be considered less often (as a low-frequency change) to enable flexibility of real-time interaction within that context. Another problem is that if there is a great deal of additional context information, the computational costs of the brute-force Puppet Master system could be problematic; potential solutions include using approximate nearest neighbor techniques (since Puppet Master uses Euclidean distance).

6.6. Wider Behavior-Creation Framework

In our work we briefly touched on the question of how Puppet-Master-like SBD may be useful for programmers. We would like further to explore how SBD will integrate into a broader behavior-creation system or existing behavior models, for example, when using SBD to show a robot how to follow a person on a street the robot can have an existing goal-based algorithm for following and avoiding objects; SBD can focus on movement texture only.

Similarly, when incorporating professional programmers and a bigger behavior context, we need to explore what practical roles SBD may be able to provide in production, for example, as an early prototyping tool, or perhaps some Puppet Master parameters should be exposed to programmers to enable them to tweak a behavior.

6.7. Definition of Style

Throughout our work we focus on the high-frequency, detailed components of motion as the important style elements. Future work should consider what limitations are introduced by this definition, and explore other definitions. For example, Laban Motion Analysis [Newlove and Dalby 2003] is a comprehensive framework for describing human motion, including subjective elements such as intentionality; could motion be distilled into these features and those then used to drive Puppet Master?

6.8. Alternate Directions

In contradiction to the overall approach presented in this paper we still believe it is important to pursue alternate directions. For example, while we aimed for a general-purpose style algorithm (within the context of texture-based movement style), it may make sense to special-tailor the approach to narrower scenarios or even a particular behavior, where context-specific optimizations or assumptions can be introduced. The same may apply for interaction design: a risk with our free-form approach is making a general interface which can do many things reasonably well but may not perform as well as a specific targeted one. Would it make sense to consider one specific interface for demonstrating a *stalker* and one for *burglar*, each with context-specific adaptions?

Similarly, while we shy away from the stimulus-response approach to defining a behavior [Wolber 1997] in our work, it may be useful to consider how such an approach would integrate with our free-form method. For example, given a scenario where one may want an explicit mechanic (such as with learning concrete rules of a dance), could Puppet Master be integrated into this with the more relaxed style elements, perhaps by analyzing demonstrations for stimulus-response and presenting them to the user.

7. CONCLUSION

As robots begin to permeate people's everyday environments, and people increasingly find themselves working with robots, it will be important for these people to be able to direct and teach these robots how they want particular tasks done. In addition to the clear goal of completing a task, the Style-by-Demonstration (SBD) approach enables people to teach robots the style of *how* the task should be done, for example, *how* a robot should shake a hand or how it should sneak around a sleeping person.

In this paper we have presented the idea of SBD, detailed three interaction and interface designs, and gave an overview of a series of studies investigating users' experiences with SBD. We provided an analysis of the various projects and highlighted the various challenges and techniques explored, and solutions implemented. In addition, we presented the Puppet Master algorithm in full detail.

Our results support our primary research hypotheses that: people naturally understand and can use SBD to create interactive robot behaviors, our interfaces are usable and enable people to demonstrate such behaviors, and the underlying Puppet Master algorithm is effective in generating satisfactory learning and mimicry results that people can understand.

REFERENCES

Jacopo Aleotti, Stefano Caselli, and Giuliano Maccherozzi. 2005. Trajectory Reconstruction with NURBS Curves for Robot Programming by Demonstration. In *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation*. IEEE Computer Society, IEEE Computer Society Press, Los Alamitos, CA, USA.

Jeff Allen, James E Young, Daisuke Sakamoto, and Takei Igarashi. 2012. Style by Demonstration for Interactive Robot Motion. In *Proceedings of the conference on Designing Interactive Systems, 2012. DIS '12, Newcastle, UK, June 11–15, 2012*. ACM, ACM Press, New York, NY, USA.

Kenji Amaya, Armin Bruderlin, and Tom Calvert. 1996. Emotion from motion. In *Proceedings of Graphics Interface 1996. GI '96, Toronto, Canada, May 22–24, 1996*. Canadian Human-Computer Communications Society, Vancouver, CANADA, 222–229.

Nalini Ambady and Robert Rosenthal. 1992. Thin slices of expressive behavior as predictors of interpersonal consequences: A meta-analysis. *Psychological Bulletin* 111, 2 (mar 1992), 256–274.

Christoph Bartneck and Jodi Forlizzi. 2004. A Design-Centred Framework for Social Human-Robot Interaction. In *Proceedings of the IEEE International Workshop on Robot and Human Interactive Communication, 2004. RO-MAN '04, Kurashiki, Okayama, Japan, September 20–22, 2004*. IEEE Computer Society, IEEE Computer Society Press, Los Alamitos, CA, USA, 581–594. DOI:http://dx.doi.org/10.1109/ROMAN.2004.1374827

Christoph Bartneck, Marcel Verbunt, Omar Mubin, and Abdullah Al Mahmud. 2007. To kill a mockingbird robot. In *Proceedings of the 2nd ACM/IEEE Conference on Human-Robot Interaction, 2007. HRI '07, Washington, D.C.,* *USA, March 10–12, 2007*. ACM, ACM Press, New York, NY, USA, 81–87. DOI:http://dx.doi.org/10.1145/1228716.1228728

H. Russel Bernard. 2000. *Social Research Methods: Qualitative and Quantitative Approaches*. Sage Publications, Thousand Oaks, CA, USA.

Cindy L. Bethel and Robin R. Murphy. 2010. Non-Facial and Non-Verbal Affective Expression for Appearance-Constrained Robots used in Victim Management. *International Journal of Behavioral Robotics (IJBR)* 1, 4 (2010), 219–230. DOI:http://dx.doi.org/10.2478/s13230-011-0009-5

Bruce M. Blumberg and Tinsley A. Galyean. 1995. Multi-Level Direction of Autonomous Creatures for Real-Time Virtual Environments. In *Proceedings of the ACM SIGGRAPH International Conference and Exhibition on Computer Graphics and Interactive Techniques, 1995. SIGGRAPH '95, Los Angeles, CA, USA, August 6–11, 1995*. ACM, ACM Press, New York, NY, USA, 47–54. DOI:http://dx.doi.org/10.1145/218380.218405

Cynthia L. Breazeal. 2002. *Designing Sociable Robots*. The MIT Press, Cambridge, MA, USA.

Cynthia L. Breazeal, Andrew G. Brooks, Jesse Gray, Guy Hoffman, Cory D. Kidd, Hans Lee, Jeff Lieberman, Andrea Lockerd, and David Mulanda. 2004. Tutelage and collaboration for humanoid robots. *International Journal of Humanoid Robotics (IJHR)* 1, 2 (2004), 315–348. DOI:http://dx.doi.org/10.1142/S0219843604000150

Cynthia L. Breazeal and Brian Scassellati. 1999. A context-dependent attention system for a social robot. In *Proceedings of the International Joint Conference on Artificial Ingelligence, 1999. IJCAI '99, Stockholm, Sweden, July 33–August*. Morgan Kauffman Publishers, imprint of Elsevier, San Francisco, CA, USA, 1146–1151.

Emily A. Butler, Boris Egloff, Frank H. Wilhelm, Nancy C. Smith, Elizabeth A. Erickson, and James J. Gross. 2003. The social consequences of expressive suppression. *Emotion* 3, 1 (2003), 48–67.

Daniel Byers and Odest C. Jenkins. 2008. HRI Caught on Film 2: Hands-free Human-Robot Interaction. In *Proceedings of the 3rd ACM/IEEE Conference on Human-Robot Interaction, 2008. HRI '08, Amsterdam, The Netherlands, March 12–15, 2008*, Christoph Bartneck (Ed.). ACM, ACM Press, New York, NY, USA. DOI:http://dx.doi.org/10.1145/1349822.1349873

Michael Chueh, Sanjay Joshi, Yi Lin Au Yeung, and Kim-Pang Lei. 2006. Towards a Mobile-Robot Following Controller Using Behavioral Cues. In *Proceedings of the IEEE International Conference on Industrial Technology, 2006. ICIT '06, Mumbai, India, December 15–17, 2006*. IEEE Computer Society, IEEE Computer Society Press, Los Alamitos, CA, USA, 150–157. DOI:http://dx.doi.org/10.1109/ICIT.2006.372317

Allen Cypher. 1991. EAGER: programming repetitive tasks by example. In *Proceedings of the ACM Conference on Human Factors in Computing Sysems, 1991. CHI '91, New Orleans, Louisiana, USA, April 27–May 02, 1991*. ACM, ACM Press, New York, NY, USA, 33–39. DOI:http://dx.doi.org/10.1145/108844.108850

Kirsten Dautenhahn. 2002. Design spaces and niche spaces of believable social robots. In *Proceedings of the IEEE International Workshop on Robot and Human Interactive Communication, 2002. ROMAN '02, Berlin, Germany, September 25–27, 2002*. IEEE Computer Society, IEEE Computer Society Press, Los Alamitos, CA, USA, 192–197. DOI:http://dx.doi.org/10.1109/ROMAN.2002.1045621

Rüdiger Dillmann. 2004. Teaching and learning of robot tasks via observation of human performance. *Robotics and Autonomous Systems* 47, 2–3 (June 2004), 109–116. DOI:http://dx.doi.org/10.1016/j.robot.2004.03.005

Jonathan Dinerstein, Parris K. Egbert, and Dan Ventura. 2007. Learning Policies for Embodied Virtual Agents through Demonstration. In *Proceedings of 20th Interna-*

tional Joint Conference on Artificial Intelligence, 2007. IJCAI '07, Hyderabad, India, January 6–12, 2007. IJCAI, IJCAI Press, Rochester Hills, MI, USA, 1257–1262.

Mira Dontcheva, Gary Yngve, and Zoran Popović. 2003. Layered acting for character animation. *ACM Transactions on Graphics* 22, 3 (July 2003), 409–416. DOI:http://dx.doi.org/10.1145/882262.882285

Ylva Fernaeus, Sara Ljungblad, Mattias Jacobsson, and Alex Taylor. 2009. Where Third Wave HCI Meets HRI: report from a workshop on user-centred design of robots. In *adjunct proceedings of the ACM/IEEE International Conference on Human-Robot Interaction (Late-Breaking Abstracts), 2009. HRI LBA '09, San Diego, California, USA, March 11–13, 2009*. ACM, ACM Press, New York, NY, USA, 293–294. DOI:http://dx.doi.org/10.1145/1514095.1514182

Morten Fjeld, Martin Bichsel, and Matthias Rauterberg. 1986. BUILD-IT: a brick-based tool for direct interaction. *Engineering, Psychology, and Cognitive Ergonomics* 4 (1986), 205–212.

Jodi Forlizzi and Carl DiSalvo. 2006. Service Robots in the Domestic Environment: A Study of the Roomba Vacuum in the Home. In *Proceedings of the 1st ACM SIGCHI/SIGART Conference on Human-Robot Interaction, 2006. HRI '06, Salt Lake City, USA, March 2–4, 2006*. ACM, ACM Press, New York, NY, USA, 258–256. DOI:http://dx.doi.org/10.1145/1121241.1121286

Phil Frei, Victor Su, Bakhtiar Mikhak, and Hiroshi Ishii. 2000. Curlybot: designing a new class of computational toys. In *Proceedings of the ACM Conference on Human Factors in Computing Sysems, 2000. CHI '00, The Hague, The Netherlands, April 1–6, 2000*. ACM, ACM Press, New York, NY, USA, 129–136. DOI:http://dx.doi.org/10.1145/332040.332416

Peggy E. Gallaher. 1992. Individual Differences in Nonverbal Behavior: Dimensions of Style. *Journal of Personality and Social Psychology* 63, 1 (1992), 133–145.

Rachel Gockley, Jodi Forlizzi, and Reid Simmons. 2006. Interactions with a moody robot. In *Proceedings of the 1st ACM SIGCHI/SIGART Conference on Human-Robot Interaction, 2006. HRI '06, Salt Lake City, USA, March 2–4, 2006*. ACM, ACM Press, New York, NY, USA, 186–193. DOI:http://dx.doi.org/10.1145/1121241.1121274

Rachel Gockley, Jodi Forlizzi, and Reid Simmons. 2007. Natural person-following behavior for social robots. In *Proceedings of the 2nd ACM/IEEE Conference on Human-Robot Interaction, 2007. HRI '07, Washington, D.C., USA, March 10–12, 2007*. ACM, ACM Press, New York, NY, USA, 17–24. DOI:http://dx.doi.org/10.1145/1228716.1228720

Elena Gribovskaya and Aude Billard. 2008. Combining dynamical systems control and programming by demonstration for teaching discrete bimanual coordination tasks to a humanoid robot. In *Proceedings of the 3rd ACM/IEEE Conference on Human-Robot Interaction, 2008. HRI '08, Amsterdam, The Netherlands, March 12–15, 2008*. ACM, ACM Press, New York, NY, USA, 33–40. DOI:http://dx.doi.org/10.1145/1349822.1349828

Dan Halbert. 1984. *Programming by Example*. Ph.D. Dissertation. University of California Berkeley.

Judith A. Hall and Erick J. Coats. 2005. Nonverbal Behavior and the Vertical Dimension of Social Relations: A Meta-Analysis. *Psychological Bulletin* 131, 6 (2005), 898–924.

John Harris and Ehud Sharlin. 2011. Exploring the Affect of Abstract Motion in Social Human-Robot Interaction. In *Proceedings of the IEEE International Workshop on Robot and Human Interactive Communication, 2011. ROMAN '11, Atlanta, US, July 31 – Aug 3, 2011*. IEEE Computer Society, IEEE Computer Society Press, Los Alamitos, CA, USA.

Fritz Heider and Marianne Simmel. 1944. An Experimental Study of Apparent Behavior. *American Journal of Psychology* 57 (1944), 243–259.

Alexis Heloir, Michael Kipp, Sylvie Gibet, and Nicolas Courty. 2008. Evaluating Data-Driven Style Transformation for Gesturing Embodied Agents. In *Intelligent Virtual Agents*, Helmut Prendinger, James Lester, and Mitsuru Ishizuka (Eds.). Lecture Notes in Computer Science, Vol. 5208. Springer Berlin / Heidelberg, Berlin, New York, Heidelberg, 215–222. DOI:http://dx.doi.org/10.1007/978-3-540-85483-8_22

Aaron Hertzmann, Charles E. Jacobs, Nuria Oliver, Brian Curless, and David H. Salesin. 2001. Image Analogies. In *Proceedings of the ACM SIGGRAPH International Conference and Exhibition on Computer Graphics and Interactive Techniques, 2001. SIGGRAPH '01, Los Angeles, CA, USA, August 12–17, 2001*. ACM, ACM Press, New York, NY, USA, 327–340.

Aaron Hertzmann, Nuria Oliver, Brian Curless, and Steven M. Seitz. 2002. Curve analogies. In *Proceedings of the ACM EUROGRAPHICS Workshop on Rendering, 2002. EGRW '02, Pisa, Italy, June 26–28, 2002*. ACM, ACM Press, New York, NY, USA, 233–246.

Helge Hüttenrauch and Kerstin Severinson Eklundh. 2004. Investigating socially interactive robots that give the right cues and make their presence felt. In *Proceedings of the Workshop on Shaping Human-Robot Interaction, Proceedings of the ACM Conference on Human Factors in Computing Sysems*. ACM, ACM Press, New York, NY, USA, 17–19.

Takeo Igarashi, Tomer Moscovich, and John F. Hughes. 2005. As-rigid-as-possible shape manipulation. *ACM Transactions on Graphics* 24, 3 (July 2005), 1134–1141. DOI:http://dx.doi.org/10.1145/1073204.1073323

Hiroshi Ishiguro. 2007. Android Science: Toward a New Cross-Interdisciplinary Framework. *Springer Tracts in Advanced Robotics* 28 (2007), 118–127. DOI:http://dx.doi.org/10.1007/978-3-540-48113-3_11

Hiroshi Ishii and Brygg Ullmer. 1997. Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms. In *Proceedings of the ACM Conference on Human Factors in Computing Sysems, 2007. CHI '97, Atlanta, GA, USA, March 22–27, 1997*. ACM, ACM Press, New York, NY, USA, 234–241. DOI:http://dx.doi.org/10.1145/258549.258715

Takayuki Kanda, Masayuki Kamashima, Michita Imai, Tetsuo Ono, Daisuke Sakamoto, Hiroshi Ishiguro, and Yuichiro Anzai. 2007. A humanoid robot that pretends to listen to route guidance from a human. *Autonomous Robots* 22, 1 (Jan. 2007), 87–100. DOI:http://dx.doi.org/10.1007/s10514-006-9007-6

Saul M. Kassin. 1982. Heider and Simmel (1944) revisited: causal attribution and the animated film technique. *Review of Personality and Social Psychology* 3 (1982), 125–150.

Sara Kiesler and Pamela Hinds. 2004. Introduction to This Special Issue on Human-Robot Interaction. *Human Computer Interaction* 19, 1/2 (2004), 1–8. DOI:http://dx.doi.org/10.1109/TSMCA.2005.850577

Johannes Kopf, Chi-Wing Fu, Daniel Cohen-Or, Oliver Deussen, Dani Lischinski, and Tien-Tsin Wong. 2007. Solid Texture Synthesis from 2D Exemplars. *ACM Transactions on Graphics* 26, 3 (2007), 2:1–2:9. DOI:http://dx.doi.org/10.1145/1276377.1276380

Amy LaViers and Magnus Egerstedt. 2012. Style Based Robotic Motion. In *Proceedings of the American Control Conference, Montreal*. IEEE Computer Society Press, Los Alamitos, CA, USA.

Jehee Lee and Kang Hoon Lee. 2004. Precomputing avatar behavior from human motion data. In *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation, 2004. SCA '04, Grenoble, France, August 27–29, 2004*. Eurographics Association Press, Eurographics Association, Germany, 79–87.

DOI:http://dx.doi.org/10.1145/1028523.1028535

Alon Lerner, Yiorgos Chrysanthou, and Dani Lischinski. 2007. Crowds by Example. *Computer Graphics Forum* 26, 3 (Sept. 2007), 655–664. DOI:http://dx.doi.org/10.1111/j.1467-8659.2007.01089.x

Martijn Liem, Arnoud Visser, and Frans Groen. 2008. A hybrid algorithm for tracking and following people using a robotic dog. In *Proceedings of the 3rd ACM/IEEE Conference on Human-Robot Interaction, 2008. HRI '08, Amsterdam, The Netherlands, March 12–15, 2008*. ACM, ACM Press, New York, NY, USA, 185–192. DOI:http://dx.doi.org/10.1145/1349822.1349847

Andrea Lockerd and Cynthia L. Breazeal. 2004. Tutelage and Socially Guided Robot Learning. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 2004. IROS '04, Sendai, Japan, September 28–October 2, 2004*, Vol. 4. IEEE Computer Society, IEEE Computer Society Press, Los Alamitos, CA, USA, 3475–3480. DOI:http://dx.doi.org/10.1109/IROS.2004.1389954

A. Chris Long, James A. Landay, and Lawrence A. Rowe. 2001. "Those look similar!" issues in automating gesture design advice. In *Proceedings of the Workshop on Perceptive User Interfaces, 2001. PUI '01, Orlando, Fl, USA, November 15–16*. ACM, ACM Press, New York, NY, USA, 15–16.

Pattie Maes. 1995. Artificial life meets entertainment: lifelike autonomous agents. *Commun. ACM* 38, 11 (Nov. 1995), 108–114. DOI:http://dx.doi.org/10.1145/219717.219808

Takamitsu Matsubara, Sang-Ho Hyon, and Jun Morimoto. 2010. Learning Stylistic Dynamic Movement Primitives from Multiple Demonstrations. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 2010. IROS '10, Taipei, Taiwan, October 18–22, 2010*. IEEE Computer Society Press, IEEE Computer Society Press, Los Alamitos, CA, USA, 1277–1283.

Daisuke Matsui, Takashi Minato, Karl F. MacDorman, and Hiroshi Ishiguro. 2005. Generating Natural Motion in an Android by Mapping Human Motion. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 2005. IROS '05, Edmonton, Canada, August 2–6, 2005*. IEEE Computer Society, IEEE Computer Society Press, Los Alamitos, CA, USA, 1089–1096. DOI:http://dx.doi.org/10.1109/IROS.2005.1545125

David L. Maulsby, Ian H. Witten, and Kenneth A. Kittlitz. 1989. Metamouse: specifying graphical procedures by example. In *Proceedings of the ACM SIGGRAPH International Conference and Exhibition on Computer Graphics and Interactive Techniques, 1989. SIGGRAPH '89, Boston, MA, USA, July 31 – August 4, 1989*. ACM, ACM Press, New York, NY, USA, 127–136. DOI:http://dx.doi.org/10.1145/74333.74346

Marek P. Michalowski, Selma Sabanovic, and Hideki Kozima. 2007. A dancing robot for rhythmic social interaction. In *Proceedings of the 2nd ACM/IEEE Conference on Human-Robot Interaction, 2007. HRI '07, Washington, D.C., USA, March 10–12, 2007*. ACM, ACM Press, New York, NY, USA, 89–96. DOI:http://dx.doi.org/10.1145/1228716.1228729

Yasser Mohammad, Shogo Okada, and Toyoai Nishida. 2010. Autonomous Development of Gaze Control for Natural Human-Robot Interaction. In *Adjunct Proceedings of the 2010 International Conference on Intelligent User Interfaces Workshop on Eye Gaze in Intelligent Human Machine Interaction*. ACM, ACM Press, New York, NY, USA.

Bilge Mutlu, Toshiyuki Shiwa, Takayuki Kanda, Hiroshi Ishiguro, and Norihiro Hagita. 2009. Footing in Human-Robot Conversations: How Robots Might Shape Participant Roles Using Gaze Cues. In *Proceedings of the 4th ACM/IEEE Conference on Human-Robot Interaction, 2009. HRI '09, San Diego, Califor-*

nia, USA, March 11–13, 2009. ACM, ACM Press, New York, NY, USA, 61–68. DOI:http://dx.doi.org/10.1109/ROMAN.2007.4415251

Clifford Nass and Youngme Moon. 2000. Machines and Mindlessness: Social Responses to Computers. *Journal of Social Issues* 56, 1 (2000), 81–103. DOI:http://dx.doi.org/10.1111/0022-4537.00153

Jean Newlove and John Dalby. 2003. *Laban For All*. Nick Hern Books, UK.

Nuno Otero, Aris Alissandrakis, Kerstin Dautenhahn, Chrystopher Nehaniv, Dag S. Syrdal, and Kheng Lee Koay. 2008. Human to robot demonstrations of routine home tasks: exploring the role of the robot's feedback. In *Proceedings of the 3rd ACM/IEEE Conference on Human-Robot Interaction, 2008. HRI '08, Amsterdam, The Netherlands, March 12–15, 2008*. ACM, ACM Press, New York, NY, USA, 177–184. DOI:http://dx.doi.org/10.1145/1349822.1349846

Ivan P. Pavlov. 1927. *Conditioned Reflexes: An Investigation of the Physiological Activity of the Cerebral Cortex*. Oxford University Press, Oxford, UK. Translated and Edited by G. V. Anrep.

Hayes Solos Raffle, Amanda J. Parkes, and Hiroshi Ishii. 2004. Topobo: a constructive assembly system with kinetic memory. In *Proceedings of the ACM Conference on Human Factors in Computing Sysems, 2004. CHI '04, Vienna, Austria, April 24–29, 2004*. ACM, ACM Press, New York, NY, USA, 647–654. DOI:http://dx.doi.org/10.1145/985692.985774

Byron Reeves and Clifford Nass. 1996. *The Media Equation: How people treat computers, television, and new media like real people and places* (first paperback ed.). CSLI Publications, Center for the Study of Language and Information Leland Standford Junior University, Cambridge, UK.

Craig W. Reynolds. 1987. Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the ACM SIGGRAPH International Conference and Exhibition on Computer Graphics and Interactive Techniques, 1987. SIGGRAPH '87, Anaheim, CA, USA, July 27–31, 1987*. ACM, ACM Press, New York, NY, USA, 25–34. DOI:http://dx.doi.org/10.1145/37401.37406

Martin Saerbeck and Christoph Bartneck. 2010. Perception of Affect Elicited by Robot Motion. In *Proceedings of the 5th ACM/IEEE Conference on Human-Robot Interaction, 2010. HRI '10, Osaka, Japan, March 2–5, 2010*. ACM, ACM Press, New York, NY, USA, 53–60. DOI:http://dx.doi.org/10.1145/1734454.1734473

Daisuke Sakamoto, Takayuki Kanda, Tetsuo Ono, Hiroshi Ishiguro, and Norihiro Hagita. 2007. Android as a telecommunication medium with a human-like presence. In *Proceedings of the 2nd ACM/IEEE Conference on Human-Robot Interaction, 2007. HRI '07, Washington, D.C., USA, March 10–12, 2007*. ACM, ACM Press, New York, NY, USA, 193–200. DOI:http://dx.doi.org/10.1145/1228716.1228743

Brian J. Scholl and Patrice D. Tremoulet. 2000. Perceptual causality and animacy. *Trends in Cognitive Sciences (TRENDS)* 4, 8 (Aug. 2000), 299–309. DOI:http://dx.doi.org/10.1016/S1364-6613(00)01506-0

Ehud Sharlin, Benjamin Watson, Yoshifumi Kitamura, Fumio Kishino, and Yuichi Itoh. 2004. On tangible user interfaces, humans and spatiality. *Personal and Ubiquitous Computing* 8, 5 (2004), 338–346. DOI:http://dx.doi.org/10.1007/s00779-004-0296-5

Elaine Short, Justin Hart, Michelle Vu, and Bran Scassellati. 2010. No fair!!: an interaction with a cheating robot. In *Proceedings of the 5th ACM/IEEE Conference on Human-Robot Interaction, 2010. HRI '10, Osaka, Japan, March 2–5, 2010*. ACM, ACM Press, New York, NY, USA, 219–226. DOI:http://dx.doi.org/10.1145/1734454.1734546

Candace L. Sidner, Christopher Lee, Louis-Philippe Morency, and Clifton Forlines. 2006. The effect of head-nod recognition in human-robot conversation. In *Proceed-*

ings of the 1st ACM SIGCHI/SIGART Conference on Human-Robot Interaction, 2006. HRI '06, Salt Lake City, USA, March 2–4, 2006. ACM, ACM Press, New York, NY, USA, 290–296. DOI:http://dx.doi.org/10.1145/1121241.1121291

Maria Staudte and Matthew W. Crocker. 2009. Visual attention in spoken human-robot interaction. In *Proceedings of the 4th ACM/IEEE Conference on Human-Robot Interaction, 2009. HRI '09, San Diego, California, USA, March 11–13, 2009*. ACM, ACM Press, New York, NY, USA, 77–84. DOI:http://dx.doi.org/10.1145/1514095.1514111

Anselm Strauss and Juliet Corbin. 1998. *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*. Sage Publications, Thousand Oaks, London, New Delhi.

Ja-Young Sung, Henrik I. Christensen, and Rebecca E. Grinter. 2009a. Robots in the wild: understanding long-term user. In *Proceedings of the 4th ACM/IEEE Conference on Human-Robot Interaction, 2009. HRI '09, San Diego, California, USA, March 11–13, 2009*. ACM, ACM Press, New York, NY, USA, 45–52.

Ja-Young Sung, Rebecca E. Grinter, and Henrik I. Christensen. 2009b. "Pimp my Roomba": designing for personalization. In *Proceedings of the ACM Conference on Human Factors in Computing Sysems, 2009. CHI '09, Boston, USA, April 4–9, 2009*. ACM, ACM Press, New York, NY, USA, 193–196. DOI:http://dx.doi.org/10.1145/1518701.1518732

Ja-Young Sung, Lan Guo, Rebecca E. Grinter, and Henrik I. Christensen. 2007. "My Roomba is Rambo": Intimate Home Appliances. In *Proceedings of the International Conference on Ubiquitous Computing, 2007. UBICOMP '07, Innbruck, Austria, September 16–17, 2007*. Lecture Notes in Computer Science, Vol. 4717/2007. Springer-Verlag, Berlin, New York, Heidelberg, 145–162. DOI:http://dx.doi.org/10.1007/978-3-540-74853-3_9

Frank Thomas and Ollie Johnston. 1981. *The Illusion of Life: Disney Animation* (first hyperion ed. ed.). Disney Editions (Walt Disney Productions), New York.

Matthew Thorne, David Burke, and Michiel van de Panne. 2004. Motion doodles: an interface for sketching character motion. In *Proceedings of the ACM SIGGRAPH International Conference and Exhibition on Computer Graphics and Interactive Techniques, 2004. SIGGRAPH '04, Los Angeles, CA, August 8–12, 2004*. ACM, ACM Press, New York, NY, USA, 424–431. DOI:http://dx.doi.org/10.1145/1186562.1015740

Lorenzo Torresani, Peggy Hackney, and Christoph Bregler. 2006. Learning Motion Style Synthesis from Perceptual Observations. In *Adjunct Proceedings of the Neural Information Processing Systems Foundation Conference, 2006. NIPS '06, Vancouver, Canada, December 4–7 (poster session)*. Neural Information Processing Systems Foundation, Curran Associates Inc., US.

Patrice D. Tremoulet and Jacob Feldman. 2000. Perception of Animacy from the Motion of a Single Object. *Perception* 29, 8 (May 2000), 943–951. DOI:http://dx.doi.org/10.1068/p3101

Aleksandar Vakanski, Iraj Mantegh, Andrew Irish, and Farrokh Janabi-Sharifi. 2012. Trajectory Learning for Robot Programming by Demonstration Using Hidden Markov Model and Dynamic Time Warping. *IEEE Transactions on Systems, Man, and Cybernetics* 42, 4 (Aug. 2012), 1039 –1052. DOI:http://dx.doi.org/10.1109/TSMCB.2012.2185694

Amy Voida, Ellie Harmon, and Ban Al-Ani. 2011. Homebrew Databases: Complexities of Everyday Information Management in Nonprofit Organizations. In *Proceedings of the ACM Conference on Human Factors in Computing Sysems, 2011. CHI '11, Vancouver, Canada, May 7–12, 2011*. ACM, ACM Press, New York, NY, USA.

Mikael Wahlström, antti Salovaara, Leena Salo, and Antti Oulasvirta. 2011. Resolving Safety-Critical Incidents in a Rally Control Center. *Human Computer Interaction* 26

(2011), 9–37.

Douglas J. Wiley and James K. Hahn. 1997. Interpolation Synthesis of Articulated Figure Motion. *IEEE Computer Graphics and Applications* 17, 6 (Nov. 1997), 39–45. DOI:http://dx.doi.org/10.1109/38.626968

Jacob O. Wobbrock, Htet Htet Aung, Brandon Rothrock, and Brad A. Myers. 2005. Maximizing the Guessability of Symbolic Input. In *Conference Abstracts and Applications, Proceedings of the ACM Conference on Human Factors in Computing Sysems, 2005. CHI '05, Portland, OR, USA, April 2–7, 2005*. ACM, ACM Press, New York, NY, USA, 1869–1872.

David Wolber. 1997. Pavlov: an interface builder for designing animated interfaces. *ACM Transactions on Computer-Human Interaction (TOCHI)* 4, 4 (1997), 347–386. DOI:http://dx.doi.org/10.1145/267135.267142

Fumitaka Yamaoka, Takayuki Kanda, Hiroshi Ishiguro, and Norihiro Hagita. 2008. How close?: model of proximity control for information-presenting robots. In *Proceedings of the 3rd ACM/IEEE Conference on Human-Robot Interaction, 2008. HRI '08, Amsterdam, The Netherlands, March 12–15, 2008*. ACM, ACM Press, New York, NY, USA, 137–144. DOI:http://dx.doi.org/10.1145/1349822.1349841

James E. Young, Richard Hawkins, Ehud Sharlin, and Takeo Igarashi. 2009. Toward Acceptable Domestic Robots: Applying Insights from Social Psychology. *Inagural Issue of the International Journal on Social Robotics.* 1, 1 (Jan. 2009), 95–108. DOI:http://dx.doi.org/10.1007/s12369-008-0006-y

James E. Young, Takeo Igarashi, and Ehud Sharlin. 2008. Puppet Master: Designing Reactive Character Behavior by Demonstration. In *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation, 2008. SCA '08, Dublin, Ireland, July 7–9, 2008*. Eurographics Association Press, Eurographics Association, Germany, 183–191.

James E. Young, Kentaro Ishii, Takeo Igarashi, and Ehud Sharlin. 2010. Style-by-demonstration: Using broomsticks and Tangibles to Show Robots How To Follow People. In *adjunct proceedings of the ACM/IEEE International Conference on Human-Robot Interaction (Late-Breaking Abstracts), 2010. HRI LBA '10, Osaka, Japan, March 2–5, 2010*. ACM, ACM Press, New York, NY, USA.

James E Young, Kentaro Ishii, Takeo Igarashi, and Ehud Sharlin. 2012. Style by Demonstration: Teaching Interactive Movement Style to Robots. In *Proceedings of the international conference on Intelligent User Interfaces, 2012. IUI '12, Lisbon, Portugal, February 14–17, 2012*. ACM, ACM Press, New York, NY, USA.

James E Young, JaYoung Sung, Amy Voida, Ehud Sharlin, Takeo Igarashi, Henrik Christensen, and Rebecca Grinter. 2010. Evaluating Human-Robot Interaction: Focusing on the Holistic Interaction Experience. *International Journal on Social Robotics* 3, 1 (2010), 53–67.

Received March 2012; revised a; accepted b