Style by Demonstration: Using Broomsticks and Tangibles to Show Robots How to Follow People

James E Young[1,3], Kentaro Ishii[2,3], Takeo Igarashi[2,3], Ehud Sharlin[1]

[1]University of Calgary
Calgary, AB, Canada

[2]Design Interface Project
JST ERATO

[3]The University of Tokyo
Tokyo, Japan

ABSTRACT

The style in which a robot moves, including its gait or locomotion style, can project strong messages, for example, it can be easy to distinguish a happy dog from an aggressive dog simply by how it is moving, and one can often tell if a colleague is stressed simply by the way they are walking.

Defining the real-time interactive, stylistic aspects of robotic movements via programming can be difficult and time consuming. Instead, we propose to enable people to use their existing teaching skills to directly demonstrate to robots the desired *style* of robot movements; in this paper we present an initial *style-by-demonstration* (SBD) proof-of-concept that focuses on teaching a robot specific, interactive locomotion styles. We present a novel broomstick-robot interface for directly demonstrating locomotion style to a robot, and a design critique by experienced programmers that compares the designing of interactive, stylistic robotic locomotion by our Style-By-Demonstration (SBD) approach with traditional programming methods.

Categories and Subject Descriptors

H.5.2 **[Information Systems]:**User Interfaces – Interaction Styles

General Terms

Human Factors

Keywords

Programming by demonstration, tangible user interfaces, human-robot interaction, style by demonstration, locomotion style.

1. INTRODUCTION

When designing robots that enter people's everyday environments, we argue that it is important to look beyond utility and functionality and to explicitly consider elements of style. People care a great deal about the style of objects and technologies that they possess [35] (design is crucial to user experience and satisfaction [27, 35]), and we argue that they will desire their robots to be stylish, attractive and pleasing the same as they want an attractive table, wristwatch, or car. This perspective on robot design adds an important style layer to more-traditional utility and goal oriented approaches.

There is a strong body of previous research that shows how simple motion patterns lead people to construe and construct intricate stories, personalities, and emotions (e.g., [17]), and thus we point out the importance of considering the style of a robot's locomotion movement. In this paper we present a method that enables people to directly design the style of a robot's interactive locomotion, that is, the way that the robot moves around space in real-time reaction to a counterpart entity such as a person. This *stylistic locomotion* layer of communication can provide the robot with a sort of body language, for example, to inform people on how to interact with it: a robot can present itself as being confident or unsure of its commands with upbeat or hesitant movements respectively, a dangerous robot can warn people to keep away by acting aggressively, or a servant robot can inform people of its aspiration to serve by humbly walking behind them. We also believe that such elements of style will be important for enabling robots to act in a manner that would be perceived by people as being socially appropriate.

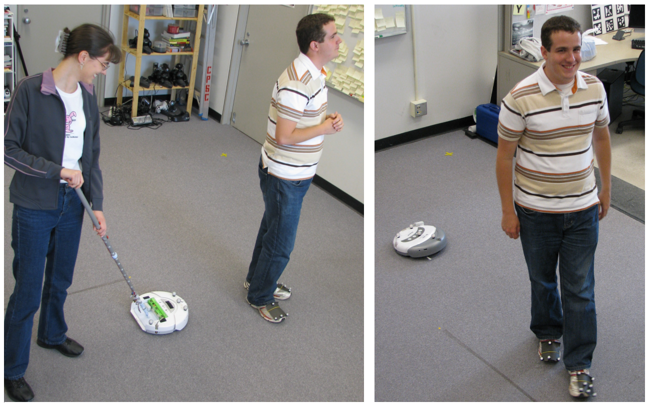
Figure 1 – Style by demonstration: a person (left) shows the robot the style of how to interact with a person (e.g., follow politely), and (right) the robot mimics the demonstration to follow politely.

Advanced programming is generally required to get robots to do even simple tasks in the real world (e.g., pick up objects, shake your hand, or follow you), in part because robot behaviors are often un-scripted and must interactively respond to dynamic environments and people's unpredictable actions in real time. Further, the style of the resulting actions tends to be very mechanical in reflection of both robots' and programming's intrinsic task-completion- and efficiency-oriented approach. To create robots that not only perform their actions but do them in, for example, an aggressive, timid, or careful fashion further complicates the programming problem.

We propose a technique we call style-by-demonstration (SBD) to bypass traditional programming and enable people to directly design the style of robot interaction by providing an exemplar. Teaching in general is a technique that people are intimately familiar with from everyday interaction with other people (i.e., from the *social stock of knowledge* [3]), and as such can be a powerful method for showing a robot the style of how to perform a task. Thus we propose that robots take advantage of these existing teaching skills to empower people and to make the difficult task of programming the style of robotic interaction accessible to them.

In this paper we present a SBD system which we designed to enable people to teach a robot how it should interact with a person, specifically, to teach the *interactive style* of the robot's *locomotion*, including a robot-on-a-broomstick tangible interface (Figure 1). We further present a non-trivial extension to the animation-only Puppet Master SBD system [36] to enable the application to robots, and finish the paper by presenting a design critique performed by experienced programmers who compared our SBD systems with traditional programming techniques. Overall, our efforts leverage the fact that human-robot interaction happens within greater real-world social contexts [10], using people's existing intuitive understanding of tangible interaction [19] and teaching ability, and the attention people place on style [27]. We believe that our work will benefit the HRI community by exploring and highlighting how robots can leverage the style of their interactive locomotion as a core component of interaction with people.

2. RELATED WORK

There has been extensive work that shows how people attribute simple geometric shapes such as triangles, circles, and points with intentionality, personalities and gender (e.g., as an aggressive male, a triangle, trying to corner a female, a circle [17]) based solely on the style of their movements around a screen [17, 21, 30, 33]. Much of Disney's success at creating believable characters has been attributed to their realization that a character's movement style is critically important [32]. This has further been evident for interactive animated characters [2, 9] and even for robots, e.g., scripted actions such as "pick up a glass" or "knock on a door" can be made to be "neutral," "shy," or "angry" based on the style in which they were performed [1]. Our work is, as far as we know, a first attempt to explore how such style can be embedded within a robot's locomotion path similar to how it is embedded in animation. The focus of our work is on how this style can be realized through direct real-time interaction with a person's movements.

A related area of robot work is that which considers robot behavior in proximity to people. Although much of this focuses on the practical engineering mobility and vision challenges [6, 23] or the development of goal-centric predictive models [7], others have looked at style-related aspects such as how close a robot should be to a person [34] and how to make following natural [12]; in this case, copying a path versus shortest route. We believe that our work is unique in its consideration of the broader case of a robot conveying style through locomotion path, and in enabling people to directly design the style of robotic motions by demonstration.

For the creation of interactive behavior, it is common to explicitly program the behavior model to define how the entity [4] or robot [5, 20] should act in given situations. This requires the difficult task of defining stylistic robot actions and behaviors in algorithmic terms, generally inaccessible to the very non-technical end-user people who will want to customize their robots' behaviors. Real-time behavior can also be synthesized from an example database [9, 22], although such approaches generally require large amounts of pre-processing (often technical-user assisted), and primarily target plausibility of physical motion such as realistic walking or collision avoidance, rather than achieving the stylistic properties of these actions in highly interactive and dynamic instances.

Programming by demonstration is a mature field of research that has been applied to such applications as programming GUI operations [26] and animation, for example, for interactive collision avoidance and planning [8], or for the direct control or static playback of animations [18]. For robots, similar approaches are used to teach navigation routes [20], how to move or pose in a human-like fashion [25] (by copying, not learning interactive behavior), or how to perform specific physical tasks [13, 16, 24, 28]. Our work is unique in that we focus on the style of interaction in the specific case of robots, rather than on a particular task goal or movement.

Some programming-by-demonstration robot systems [11, 29] enable the creation of very expressive, stylistic motions, but these result in static movements only and are not interactive to a person's input. Other work incorporates pre-scripted style and emotion into their interfaces [24], but these are statically used to represent algorithmic states such as not understanding a command and are not learned from demonstration, and so is a different usage and problem from the more general case of teaching interactive style. While programming by demonstration is popular for both animation and robotics, we posit that we are the first to focus on interactive style rather than learning a task-specific goal or a simple static motion pattern.

3. STYLE BY DEMONSTRATION

Our implementation of interactive SBD has two phases: *demonstration* and *generation*. During demonstration, an example paired motion is provided of the robot interacting with the person in the desired style. Both paths are given simultaneously in real time: the person's path is needed in addition to the robot's path to serve as an exemplar of what the robot should react to.

After demonstration, the generation phase learns the demonstrated style (the robot's reactionary characteristics) and incorporates it into real-time interaction: the person moves freely while new, appropriately styled, robot actions are generated on the fly in response. Our implementation allows demonstration to be relatively short, from 30s to 2 minutes, and generation occurs immediately with no need for pre-processing). We realize SBD for robots using non-trivial extensions to the animation-only Puppet Master algorithm [36], detailed later below.

One such extension is the addition of discrete robot actions, in this case, enabling the robot to make a *happy* or an *unhappy* sound. This functionality was added to explore the scalability of SBD (and the underlying Puppet Master algorithm) to robotic actions beyond movement. If sounds are successful, then we expect that so would other pre-packaged actions such as picking up an object or taking a photo.

3.1 A Broomstick Interface for Teaching Movement Style

Our primary goal in developing interfaces for enabling demonstration to a robot was to enable the person to focus on the motion style rather than on the mechanics of moving the robot. In this case, the robot we use is an iRobot Roomba (Figure 2), a disc-like vacuum robot that moves via a 2-wheel-plus-caster base (it cannot move sideways) and sits close to the ground. We ruled out directly tracking a person's natural movements for demonstration

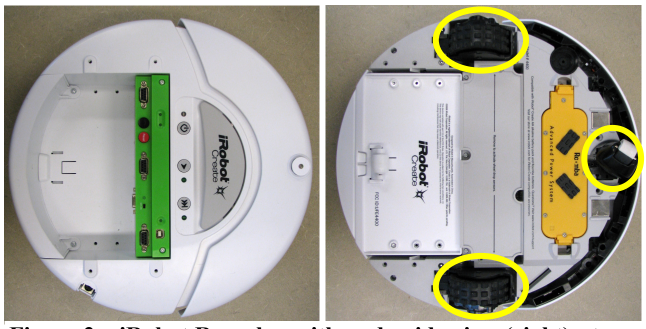
Figure 2 – iRobot Roomba, with underside view (right) : two-wheel base and coaster highlighted

as they would likely be too expressive and contain motions and nuances not reproducible by the robot. Further, the small size and height of our robot makes it impractical for a person to directly grab and show how to move.

We developed an easy-to-use broomstick tangible-user interface (TUI) that encapsulates the robot's movement properties and physical constraints. This ensures that the motions given are reproducible by the robot and further communicates the mechanical robot-movement constraints to the person. We take the TUI approach to leverage people's understanding of the tangible world and to provide immediate tactile feedback on input, thus reducing the indirection between the input and output and improving usability [19, 31]. TUIs have further been shown to afford and mediate robotic interaction; robots are arguably TUIs that themselves can sense and react in the physical world [14].

We use a regular aluminum broomstick attached to the robot (Figure 1, Figure 3) via a two-axis swivel, allowing the stick to be freely moved up, down, left and right, but not to be twisted. A user can turn the robot by tilting the broomstick left or right, or twisting it, during movement. Further, we have installed two soft-press buttons (Figure 3) on the broomstick so that the person can trigger the robot sounds (happy or sad) during demonstration. The wheels of the broomstick-Roomba have been disconnected (gears removed) to reduce the friction of the wheels and the force required to push the robot.

The result is a natural and familiar mechanism for demonstrating interactive locomotion style to the robot. In this scenario, one person (primary entity) walks naturally in the space while another person uses the broomstick to demonstrate a following style to the robot, the reactionary entity (Figure 1, left side). When demonstration is finished, a real (non-broomstick) robot (a separate Roomba) enters the space and follows the person in the manner that was demonstrated (Figure 1, right side).

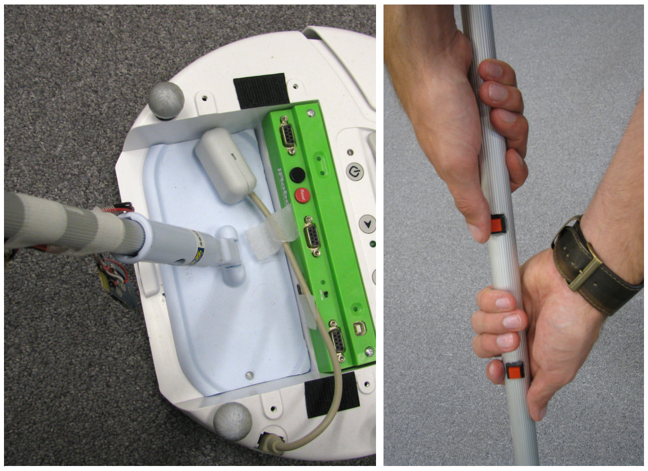
Figure 3 – broomstick-Roomba interface, with attached swivel (left) and broomstick-mounted buttons

Both robots are controlled remotely via bluetooth, and the action buttons on the broomstick are wired into a modified Phidget Remote wireless FOB clicker and standard Phidget interface kit. Both the robot's and person's locations were tracked using Vicon camera motion-tracking (Figure 4). IR-reflective markers were affixed to the robot and person, with the person's location approximated as the mid-point between shoes.

4. ALGORITHM

We realized SBD for interactive locomotion by an extension of the Puppet Master animation system [36], originally applied to generic free-form input and on-screen animation output. The primary technical challenge of this paper was to effectively apply the flexible animation-oriented Puppet Master to the hard limitations of real robots.

4.1 Original Puppet Master Algorithm

Puppet Master is an interactive-locomotion SBD algorithm that focuses on the real-time movement relationships between two animated characters. Puppet master works primarily on the relative entity positions, orientations, and velocities, examined over a time window. It uses a complex pattern-matching and generation algorithm to create real-time output (40 Hz in animation), and the algorithm's veracity is backed up by an in-depth multi-part study: satisfactory results were obtained with roughly ~30s in the animation case [36].

4.2 Adaption to Robots

Robots are imperfect physical machines that work on irregular surfaces and must adhere to real constraints such as movement speed or physical design. They cannot be moved directly (and freely) as with animated characters, and it is generally difficult for the robot to reach a Puppet-Master specified target state within the short time frame before the next target is given in response to real-

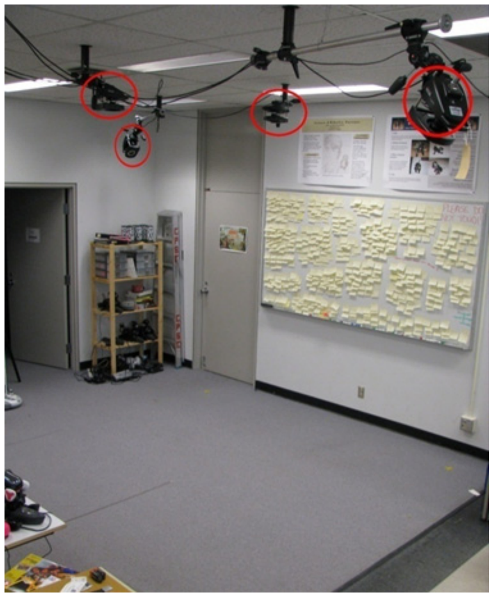
Figure 4 – the interaction space, with some of the motion-tracking cameras in view (highlighted)

time human movement; we reduced Puppet Master to 20 Hz to better reflect robot limitations. Our solution to these problems uses a translation layer between the Puppet Master algorithm output and the robot, consisting of a kinematic model, directly driving the robot with the movement texture, and using simple frequency analysis to maintain robot localization.

The kinematic model of our robot (iRobot Roomba) describes the robot's movement capabilities and constraints (Figure 5), and we use this to convert puppet master commands to something the robot can perform (and produce the actual robot commands). While this in itself is a trivial step taken for many robot implementations, this was beneficial as we did not need to modify the Puppet Master algorithm, and perhaps more importantly, we discovered that we can use this approach to perform a kind of rudimentary frequency analysis.

First, we can use this model to solve for the robot command that will move the robot to a given Puppet Master target state (Figure 6b). We see this movement to constitute the low-frequency component of the desired movement, that which places the robot in the general vicinity and orientation requested by Puppet Master. However, when the robot cannot keep up to the target, the texture (or details) of the desired Puppet Master movement are lost.

Second, the model can be used to make the robot reproduce the exact path (including texture) prescribed by Puppet Master (Figure 6c), by solving for the delta movement (change in direction, location) rather than the target location. This maintains the high-frequency component, but drift in the robot's movement for various reasons means that the robot soon loses its relative localization.

We add this high-frequency detail to the above-mentioned low-frequency result by taking a weighted average of the two resulting robot commands (velocity and turning radius). A higher focus on

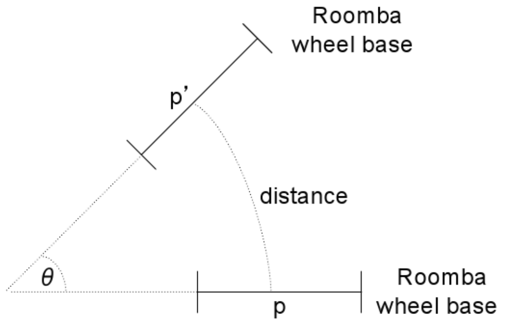

Figure 5 – Simple Roomba movement model that describes its movement constraints and properties, and how they relate to the robot's possible commands (velocity and turning radius). θ is turning radius + time, distance is velocity + time, p and p' is start and end position.

the high-frequency component results in better texture retention but more location drift, and a higher focus on the low-frequency component has the opposite effect. In our implementation we use a 70/30 high/low weight. The intuition is that existing high-frequency robot movements which tend away from the target location are modified slightly to change their direction, while movements that already tend toward the overall state are generally unchanged. Figure 7 details the dataflow process.

4.3 Auxiliary Actions

Our implementation extended the Puppet Master algorithm [36] to enable the demonstration of discrete actions (i.e., robot sounds), which can be specified by the trainer in-situ during training. We accomplished this by directly associating the demonstrated actions with the training data in parallel to the time axis. During generation, as particular training data is used in output construction, the associated (by time) action triggers are included

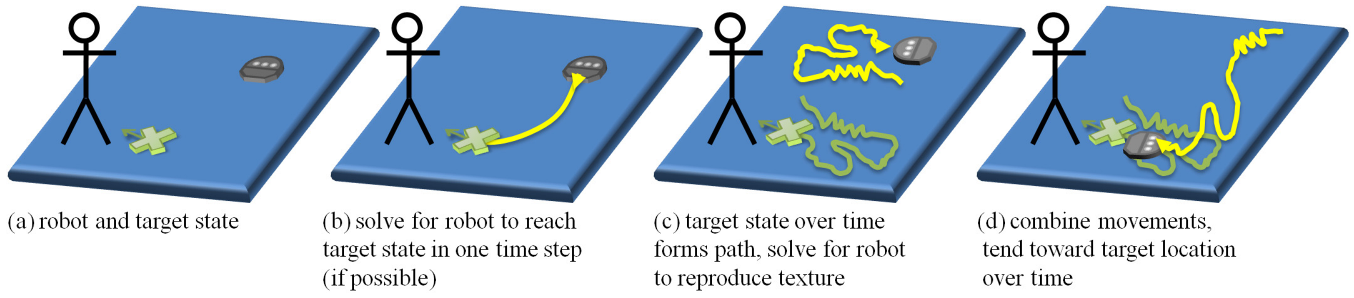
(a) robot and target state
(b) solve for robot to reach target state in one time step (if possible)
(c) target state over time forms path, solve for robot to reproduce texture
(d) combine movements, tend toward target location over time

Figure 6 – the core intuition behind our use of frequency analysis

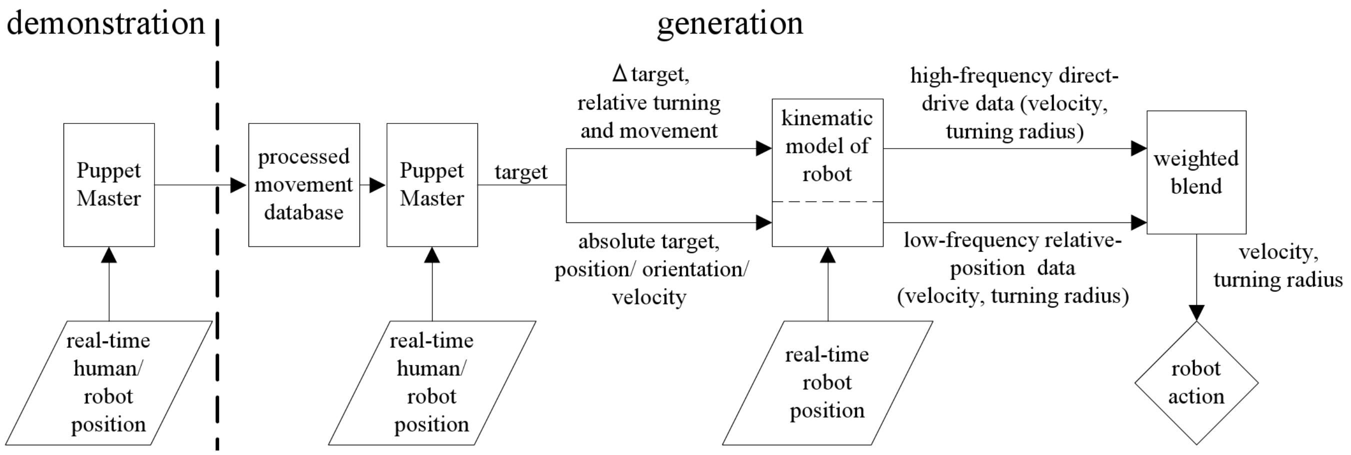
demonstration
generation
Puppet Master
real-time human/ robot position
processed movement database
Puppet Master
target
real-time human/ robot position
Δ target, relative turning and movement
absolute target, position/ orientation/ velocity
kinematic model of robot
real-time robot position
high-frequency direct-drive data (velocity, turning radius)
low-frequency relative-position data (velocity, turning radius)
weighted blend
velocity, turning radius
robot action

Figure 7 – the data-flow and processing used to adapt the output from Puppet Master [33] to work with robots

in the target output and then performed by the robot.

Our current application of robot sounds (demonstrated through button presses) serves as an important proof of concept that the Puppet Master algorithm is extendible to and can mesh with actions that are not derivative of motion paths. Robot sounds could be replaced by any discrete pre-programmed robot behavior, for example, pre-configured facial expressions, gestures, or speech comments. While this addition has not been formally tested, the preliminary results from our own experiences have been positive.

5. EXPERIENCED PROGRAMMERS DESIGN CRITIQUE

We recruited four experienced programmers from our graduate-student lab at the University of Calgary (who did not have prior exposure to our work) to create robot behaviors using Java, create the same behaviors using our broomstick SBD interface, and to reflect on their experiences with both. Our primary analysis approach is to give detailed description of the reflections to portray the thoughts and opinions of the programmers.

5.1 Study Design and Procedure

Four experienced programmers from our graduate-student lab at the University of Calgary conducted critiques. We selected four robot behaviors, a polite follow (*polite*), a robot stalking a person (*stalker*), a robot that is happy to see the person (*happy*), and a robot that is attacking a burglar (*burglar*), as variants of those used in the animated Puppet Master algorithm [36] to address reported participant concerns regarding the validity (e.g., some asked "why am I doing this?").

For the programming stage, we provided a simple robot-simulation virtual test-bench and clear API for reading the robot and person's locations and providing real-time commands. Figure 8 shows the screen view where programmers could rapidly test and fine-tune their behaviors. After initial explanation, programmers were given a total of two hours to create the four behaviors; we provided them with simple sample behaviors as a starting point. After programming, they created the same behaviors using the broomstick SBD interface.

For the broomstick SBD stage programmers were allowed to demonstrate each behavior and observe for as long as they wished before moving on to the next, and could re-train a behavior if they were not satisfied with the result. This phase was video-taped.
At the end of the study we interviewed the programmers regarding the overall experience (unstructured, video-taped).

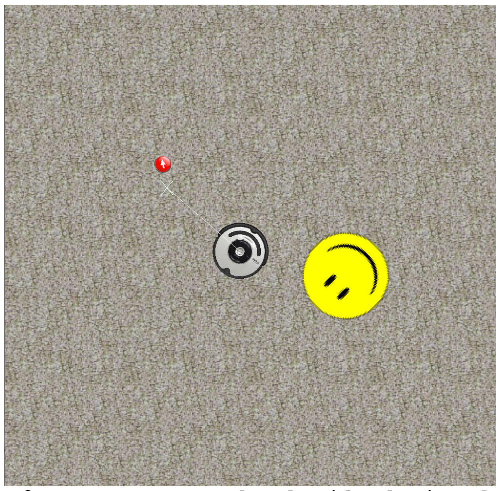
Figure 8 – programmer test-bench, with robot icon, happy-face representing the person, and X-plus-arrow for manual mouse-based robot movement

5.2 Results

All participants took the full two hours to program their behaviors, and all were able to create their behaviors in the time given, although one programmer stated that they would require a great deal more time to implement proper "nuanced behaviors." For SBD, all programmers were observed to "act the characters," making faces, laughing, etc. Figure 9 shows demonstration attempts and length (seconds). The "total time" listed is the duration of creating all behaviors, from start to finish, including observation, thinking, brief discussion, and retraining time.

When asked which they preferred, direct programming or programming by demonstration, all programmers articulated a set of trade-offs rather than preference, for example, "the programming spoke to the scientist in me, and the other, the broomstick demonstration, spoke to the non-scientific part of me."

The programming approach was touted as being more accurate and kept the person "in control" in comparison with the demonstration. Because of this, one person stated that they felt like they had "a lot more power to do something creative." However, control is not easy or complete; one programmer noted "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." The programmers made statements highlighting the difficulty of direct programming. For example: "hard to debug the program even though I have the simulated environment," "even when I program I don't know exactly what is going to happen," and "when I see problems, I still don't know why it happens." Programmers mentioned that by focusing on style the "types of things [they were] trying to express were more nuanced, more complicated behaviors" than the "easily expressed things like sine waves" that they are used to creating in interactive characters (referring to the approach to smooth animation movements). One programmer noted that doing the programming before demonstration helped to highlight the sheer difficulty of the real-time problem, and helped them to appreciate the demonstration system.

Programmers noted that the broomstick is much faster and easier than direct coding. By using the demonstration system they "did not have to think technically or analytically," and could more-easily program styles. As such "there is a huge time-saving potential here." One reason cited for the broomstick's success is

	programmer		
behavior	1	2	3
polite follow – tries	2	1	1
time	44 s	31 s	24 s
stalker – tries	1	1	1
time	65 s	46 s	31 s
attk. burglar – tries	1	1	1
time	51 s	44 s	25 s
happy to see you – tries	2	1	1
time	40 s	37 s	24 s
total time	14 m 49 s	8 m 52 s	7 m 40 s

Figure 9 – demonstration times for programmer condition; only three are shown (of four) as one participant requested not to be video-taped, and we did not record the data directly

that people are very skilled at "understanding changing situations on an instant-to-instant basis and [can] essentially make up [their] own behaviors on the fly." However, several programmers pointed out that the learning by demonstration cannot be perfect as they are "at the mercy of the system" and their "demonstration is just a small part of the bigger thing." They are "relying on its interpretations of [their] intentions, rather than on [their] actual intentions. There is no way to directly convey intentions," for example, they could not specify hard constraints such as "stay away from the corner."

We asked the programmers to give an informal design critique of our broomstick interface. One programmer mentioned that the robot can be difficult to turn quickly; however, this programmer tried to train the robot to turn much-more quickly than the real robot can perform. That is, although the interface is limited, the movement capabilities of the real robot are as well. Another, who has professional game development experience, said that although they would not consider using this as-is for something at the forefront of the game (such as a main character), they feel there is real potential for the approach for side-line characters. Another programmer pointed out that the inherent inaccuracy of the demonstration system is not necessarily a problem, as perhaps the broomstick can be used to capture basics and serve as a prototyping method for behaviors later programmed. Programmers also gave suggestions on how to mix the pure demonstration approach with more logical components, for example, to enable demonstrators to explicitly specify which components are important, or give them easy-to-understand parameters to tweak when observing the result.

Two of the programmers noted the amount of physical energy required to demonstrate using the broomstick (in the large space), that it was more physically exhausting than the act of programming. They mused about the use of a remote controller to reduce the fatigue problem, although they both admitted the result would likely be more difficult to use and control than the broomstick. One participant suggested a tabletop system as way to keep direct movement while lowering the effort required.

5.3 Discussion

This design critique helps to support the SBD idea and approach, and that it is applicable even for experienced programmers. As outlined in Figure 9, all participants managed to complete the creation and evaluation process in less than 15 minutes, substantially less than the two hours taken for programming.

What we found interesting beyond the time-efficiency results is how readily the programmers acted and *got into the characters*, laughing and making facial expressions to match what they were demonstrating. Even for scientists and engineers who have an understanding of the technical nature of the robot, our results help support the idea that programming style to robots via demonstration leverages their innate social understanding and skills, and as such, they readily accept and embrace the approach.

One of the benefits of this study was that the participants have both a technical understanding of the problem, and the *social stock of knowledge* [3] that makes the demonstration familiar and comfortable. The programmers' feedback added depth to our understanding of the accuracy/control versus time/ease trade-off. This includes such observations as programming may enable people to be creative in ways that demonstrating does not, and that the complexity of the programming approach means there is still a layer of uncertainty and mystery for experienced programmers, despite the extra control. Further, ideas were proposed on how to combine the programming (more control) and demonstration (easier to do) approaches, for example, by using the demonstration as a rapid prototyping tool, or by including easy-to-understand parameters or conditions which the demonstrator could specify or tweak.

Many of the participants' observations help us to better-understand the limitations of demonstration, for example, that there is potentially no optimal solution as machines cannot understand a person's intentions, only their actions, and this interpretation is subjective to the demonstration-learning algorithm used. We point out, however, that people suffer from the same problem as these robots: people cannot know others' intentions, only what can be deduced from interaction. Regardless, this suggests that we should aim to better understand the particular biases introduced by any given algorithm, and how this relates to target applications and usage scenarios.

The critique of the broomstick interface was very positive, supporting the interface design for manipulating a robot. The fatigue issue was a large concern, and lead to our development of a follow-up system described below.

6. ONGOING AND FUTURE WORK

One deficiency of this research to date is the lack of a formal study on the use of SBD for robots, our particular broomstick interface, as well as the quality of the Puppet Master adaption to robots. The programmed behaviors created as a by-product of this study, for example, could provide a strong comparison point against the behaviors created using our SBD implementation.

Further, in an attempt to address some of the broomstick-interface concerns raised (particularly the issue of fatigue), as well as to provide an interface comparison point, we have developed a follow-up interface based on a tabletop computer and hand-held TUI puck. We briefly describe this interface below.

6.1 Hand-Held Tabletop Puck

The hand-held tabletop puck (Figure 10) is small enough to comfortably manipulate with one hand and has a familiar mouse mounted on the top. We constructed the puck to enforce

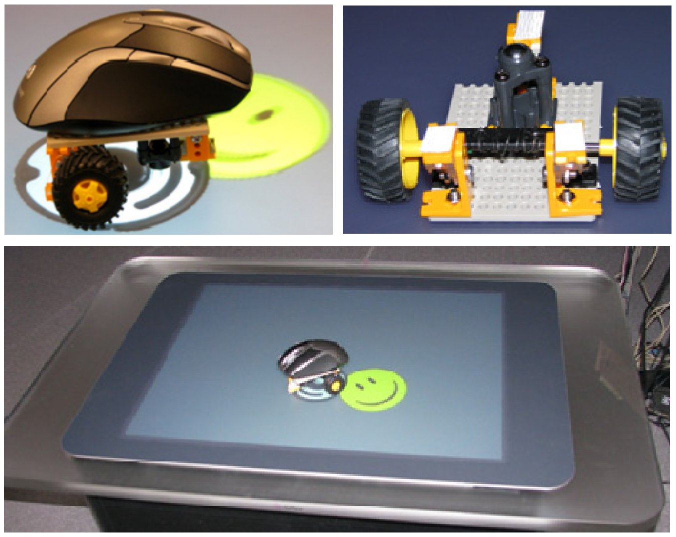
Figure 10 – hand-held tabletop puck, top and bottom view, and placed on Microsoft Surface. Notice reflective markers on the bottom for tracking

movement constraints similar to the Roomba, with two independent-axis wheels and a caster, and a slight resistance on the wheel to restrict rapid movements not reproducible by the Roomba (Figure 10). The mouse buttons on the top of the puck are used to trigger the two robot sounds (produced through speakers). The puck is placed on a Microsoft Surface, allowing a person to comfortably demonstrate style over the entire surface without fatigue, and we added reflective markers to the bottom of the puck to be detected as touches and tracked by the Surface's cameras (Figure 10). We animate a simulated (pre-scripted) person movement path shown by a *happy face* icon (Figure 10), thus enabling paired demonstration as the demonstrator uses the puck to show the robot how it should follow the happy face. After demonstration the robot can follow a person directly at the real physical space as with the broomstick, following the previously demonstrated style.

This interface design explicitly attempts to address the broomstick fatigue issue, as well as other tabletop-puck related concerns raised in previous work, for example, our markers do not hinder interaction, we constrain the TUI to reflect the robot's constraints, and lower the high cognitive load required to simultaneously demonstrate both entities [15, 36].

In the longer term we are planning to explore the scalability of both our SBD approach and our current algorithm to more elaborate robotic expressions. We are hoping that designing style by demonstration can be mapped to other applications with more complex degrees of freedom, and integrated with other robot programming approaches. For example, can we use a similar algorithm to simply demonstrate to a robotic teacher surrogate how to move its arms sternly, or enthusiastically? Can it be used to demonstrate to robotic service provider, say a waiter, how to collect dishes apologetically?

7. CONCLUSION

In this paper we presented the idea of interactive-locomotion SBD, teaching robots the *style* of how they should move to interact with people. We detailed our technical solution, introduced a novel robot-broomstick interface, and further detailed our robotic evolution of the animation-based Puppet Master SBD algorithm. Finally, we presented a design critique that helped us better understand SBD and highlighted the usefulness of the SBD approach even for experienced programmers.

With the continued advance of technology robots will be sharing more and more of our physical and social spaces. Robot acceptance will be based as much on a robot's style of interaction as on their goal oriented task performance. Providing simple, well situated tools that will enable non-technical people to show their robots in what style to act can be an important part of integrating robotic interfaces in our future everyday environments. We believe our style-by-demonstration approach presented here is an important early step in this direction.

REFERENCES

[1] Amaya, K., Bruderlin, A., and Calvert, T. Emotion from motion. In *Proceedings of Graphics Interface 1996. GI '96, Toronto, Canada, May 22–24, 1996*, 222–229.

[2] Bates, J. The role of emotion in believable agents. 122–125.

[3] Berger, P. L., and Luckmann, T. *The social construction of reality: a treatsie in the sociology of knowledge*. Anchor Books, 1966.

[4] Blumberg, B. M., and Galyean, T. A. Multi-level direction of autonomous creatures for real-time virtual environments. In *Proc. SIGGRAPH '95*, 47–54.

[5] Breazeal, C., and Scassellati, B. A context-dependent attention system for a social robot. In *Proc. IJCAI '99*, IJCAI Press (Stockholm, Sweden, 1999), 1146–1151.

[6] Byers, D., and Jenkins, O. C. HRI caught on film 2: Hands-free human-robot interaction. In *Proc. HRI '08*, ACM (NY, USA, 2008), C. Bartneck, Ed.

[7] Chueh, M., Joshi, S., Yeung, Y. L. A., and Lei, K. P. Towards a mobile-robot following controller using behavioral cues. In *Proc. ICIT '06*, IEEE, 150–157.

[8] Dinerstein, J., Egbert, P. K., and Ventura, D. Learning policies for embodied virtual agents through demonstration. In *Proc. IJCAI '07*, 1257–1262.

[9] Dontcheva, M., Yngve, G., and Popovic, Z. Layered acting for character animation. *ACM Trans. Graph. 22*, 3 (2003), 409–416.

[10] Dourish, P. *Where the Action Is: The Foundation of Embodied Interaction*. MIT Press, MA, 2001.

[11] Frei, P., Su, V., Mikhak, B., and Ishii, H. curlybot: designing a new class of computational toys. In *Proc. CHI '00*, ACM (NY, USA, 2000), 129–136.

[12] Gockley, R., Forlizzi, J., and Simmons, R. Natural person-following behavior for social robots. In *Proc. HRI '07*, ACM (NY, USA, 2007), 17–24.

[13] Gribovskaya, E., and Billard, A. D. Combining dynamical systems control and programming by demonstration for teaching discrete bimanual coordination tasks to a humanoid robot. In *Proc. HRI '08*, ACM (NY, 2008), 33–40.

[14] Guo, C., and Sharlin, E. Exploring the use of tangible user interfaces for human-robot interaction: a comparitive study. In *Proc. CHI '08*, ACM (2008), 121–130.

[15] Guo, C., Young, J. E., and Sharlin, E. Touch and toys: new techniques for interaction with a remote group of robots. In *Proc. CHI '09*, 491–500.

[16] Halbert, D. *Programming by Example*. PhD thesis, University of California Berkeley, 1984.

[17] Heider, F., and Simmel, M. An experimental study of apparent behavior. *American J. of Psychology 57* (1944), 243–259.

[18] Igarashi, T., Moscovich, T., and Hughes, J. F. As-rigid-as-possible shape manipulation. *ACM Trans. Graph. 24*, 3 (2005), 1134–1141.

[19] Ishii, H., and Ullmer, B. Tangible bits: Towards seamless interfaces between people, bits and atoms. In *Proc. CHI '97*, ACM (1997), 234–241.

[20] Kanda, T., Kamasima, M., Imai, M., Ono, T., Sakamoto, D., Ishiguro, H., and Anzai, Y. A humanoid robot that pretends to listen to route guidance from a human. *Auton. Robots 22*, 1 (2007), 87–100.

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

[22] Lee, J., and Lee, K. H. Precomputing avatar behavior from human motion data. In *Proc. SCA '04*, Eurographics

Association (Aire-la-Ville, Switzerland, Switzerland, 2004), 79–87.

[23] LIEM, M., VISSER, A., AND FROEN, G. A hybrid algorithm for tracking and following people using a robotic dog. In *Proc. HRI '08*, ACM (NY, USA, 2008), ACM, 185–192.

[24] LOCKERD, A., AND BREAZEAL, C. Tutelage and socially guided robot learning. In *Proc. IEEE/RSJ IROS '04*, vol. 4, IEEE, 3475–3480.

[25] MATSUI, D., MINATO, T., MACDORMAN, K. F., AND ISHIGURO, H. Generating natural motion in an android by mapping human motion. In *Proc. IROS '05*, ACM (NY, USA, 2005), 1089–1096.

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

[27] NORMAN, D. A. *Emotional Design: Why We Love (or Hate) Everyday Things*. Basic Books, NY, 2004.

[28] OTERO, N., ALISSANDRAKIS, A., DAUTENHAHN, K., NEHANIV, C., SYRDAL, D., AND KOAY, K. Human to robot demonstrations of routine home tasks: exploring the role of the robot's feedback. In *Proc. HRI '08*, ACM (NY, 2008), 177–184.

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

[30] SCHOLL, B. J., AND TREMOULET, P. D. Perceptual causality and animacy. *Trends in Cognitive Sciences (TRENDS) 4*, 8 (Aug. 2000), 299–309.

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

[32] THOMAS, F., AND JOHNSTON, O. *The Illusion of Life: Disney Animation*, first hyperion ed. ed. Disney Editions (Walt Disney Productions), 1981.

[33] TREMOULET, P. D., AND FELDMAN, J. Perception of animacy from the motion of a single object. 943–951.

[34] YAMAOKA, F., KANDA, T., ISHIGURO, H., AND HAGITA, N. How close?: model of proximity control for information-presenting robots. In *Proc. HRI '08*, ACM (NY, USA, 2008), ACM, 137–144.

[35] YOUNG, J. E., HAWKINS, R., SHARLIN, E., AND IGARASHI, T. Toward acceptable domestic robots: Applying insights from social psychology. *Inagural Issue of the Int. J. Social Robotics. January 2009 1*, 1 (2009).

[36] YOUNG, J. E., IGARASHI, T., AND SHARLIN, E. Puppet master: Designing reactive character behavior by demonstration. In *ACM SIGGRAPH / EG SCA '08.*, EG Association (Germany, 2008), EG Press, 183–191.