User-Centered Programming by Demonstration: Stylistic Elements of Behavior*

James E. Young,[1,2] Kentaro Ishii,[2,3] Takeo Igarashi,[2,3] Ehud Sharlin[4]
young@cs.umanitoba.ca, kenta@sakamura-lab.org, takeo@acm.org, ehud@cpsc.ucalgary.ca
[1]University of Manitoba Winnipeg, MB, Canada
[2]JST ERATO Tokyo, Japan
[3]The University of Tokyo Tokyo, Japan
[4]University of Calgary Calgary, AB, Calgary

Abstract

User-Centered Programming by Demonstration is an approach that places the needs of people above algorithmic constraints and requirements. In this paper we present a user-centered programming by demonstration project for authoring interactive robotic locomotion style. The style in which a robot moves about a space, expressed through its motions, can be used for communication. For example, a robot could move aggressively in reaction to a person's actions, or alternatively react using careful, submissive movements. We present a new demonstration interface, algorithm, and evaluation results.

1 Introduction

It is important to look beyond technical functionality when creating robots that enter into people's everyday spaces: successful robot interface design must also consider people's needs and preferences. Robotic programming by demonstration (PBD) is one such method that enables non-expert people to use their natural demonstration abilities to author robotic actions and behaviors (e.g., see [Dinerstein et al., 2006]). However, ongoing PBD research generally focuses on the yet-unsolved learning and algorithm challenges, forcing people to work within algorithmic requirements for demonstration (see, e.g., [Suay et al., 2012]). We argue that a complementary user-centered effort is needed, where the primary driver of interaction and algorithm design is user requirements; we present one such project in this paper.

People care a great deal about style, and design greatly impacts interaction experience and satisfaction [Norman, 2002; Young et al., 2008a]. Style refers not only to physical appearance, but also includes how a robot *moves*. We present a PBD system for enabling people to teach interactive locomotion style to robots, their "expressive movement" [Gallaher, 1992], for example, how to follow someone politely, or aggressively crowd them. We call this general approach style-by-demonstration (SBD) to contrast it with the more general programming by demonstration.

With this user-centered requirement in mind, we developed a robot-on-a-broomstick tangible interface (Figure 1) and novel algorithm for enabling people to teach a robot interactive locomotion style, and present the results from an experienced-programmer design critique evaluation that compared our new approach to traditional programming.

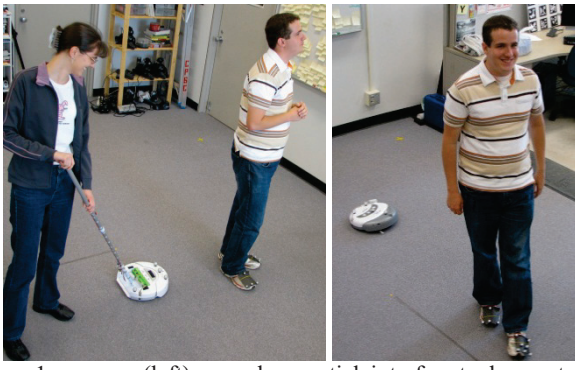


Figure 1. a person (left) uses a broomstick interface to demonstrate a particular style of interactive movement (e.g, to follow politely), and (right) the robot mimics the demonstration.

2 Related Work

People attribute simple geometric shapes such as triangles and circles with intentionality and personalities based solely on the style of their movements around a screen (for example, [Heider and Simmel, 1944; Tremoulet and Feldman, 2000]). Further, much of Disney's success at creating believable characters has been attributed to their focus on a character's movement style [Thomas and Johnston, 1995]. Interactive animation has leveraged this, for example, to perform actions such as "pick up a glass" or "knock on a door" in a "neutral," "shy," or "angry" fashion [Amaya et al., 1996]. Our work builds on these past efforts and incorporates interactive movement style into robot interaction through SBD.

Programming by demonstration has been applied to such applications as authoring GUI operations [Maulsby et al., 1989] and interactive animation or robot path planning [Dinerstein et al., 2006; Kanda et al., 2006], or for control of semi-static animations [Igarashi and Igarashi, 2009]. Other methods synthesize from large example databases, requiring extensive technical-user pre-processing [Lee and Lee, 2004]. For robots, it is more common to teach the goals of specific physical tasks [Gribovskaya and Billard, 2008]. Our work is unique in that rather than teaching the steps needed for achieving a specific goal we instead concentrate on teaching the high-level *style* of interactive behavior. Some robot systems enable the creation of stylistic and expressive motions [Frei et al., 2000; Raffle et al., 2004], but these result in static movements only and are not interactive to input. As far as we

*The paper on which this extended abstract is based was the recipient of the best paper award of the 2012 International Conference on Intelligent User Interfaces, IUI '12. [Young et al., 2012]

know there is no previous work on explicitly teaching a robot interactive style via demonstration.

3 Interaction Design

Our system enables a user to teach an iRobot Roomba robot how to interact with a person, via two phases: *demonstration* and *generation*. A user demonstrates an example of *how* (in what style) the robot should interact with a person moving about a space. For *demonstration* both robot and person path are given simultaneously to provide example person movements that the interactive robot should react to (Figure 1).

After *demonstration* the robot *generates* real-time interactive behavior using the learned reactionary features; the person moves freely as before but the robot autonomously interacts using the trained movement style (Figure 1). The entire behavior derives from the training and includes no pre-programmed movements or algorithms.

3.1 A Broomstick Interface for Teaching Style

To enable users to focus on demonstration rather than robot mechanics, we developed a broomstick attached to an iRobot Create disc robot (Figure 1). We could not simply track a person's movements as they would contain motions not reproducible by the robot. Using a robot replica further encapsulates the robot's movement properties (e.g., cannot move sideways, etc.) into a tangible interface that makes the properties immediately clear, and ensures that demonstrations are reproducible by the actual robot. The broomstick is attached by a two-axis swivel, enabling forward and backward movement as well as turning via twisting or angled pushing. After teaching using the broomstick interface, a robot enters the space for live interaction (Figure 1). The robot and broomstick are tracked using Vicon motion capture and communication happens over Bluetooth connections.

4 Algorithm for Learning Style

We extended our previous animation-only Puppet Master algorithm [Young et al., 2008b] to achieve learning. This method forms feature vectors based on the robot's location in relation to the person's (relative position, orientation, and velocity), and searches this space for training data similar to a given real-time human-robot situation (Euclidean distance). The result is used to generate movement, and the process is repeated at 20hz such that the output is a kind of patch-work of training data pieces. This is a highly simplified overview of the algorithm [Young et al., 2012].

Puppet Master emphasizes the learning and generation of movement style. Because of this, it primarily focuses on the details (or *texture*) of movement and does not include any explicit process learning. A problem with adapting Puppet Master to robots is that the prior system relied on complete freedom of movement and did not work with the hard constraints and relatively slow response times of robotic motors. In particular, while the animation method exaggerated and modified movements to maintain locality (e.g., the robot being behind the person), a real robot could not, for example, be so easily moved at double speed or nudged slightly sideways. Such movements are often necessary to compensate for drift, particularly when the robot is reproducing training regions with a great deal of movement details (e.g., shaking back and forth).

4.1 Detail and Relative-Position Components of Path Generation

Our proposed solution revolves around a hypothesis that interactive robot movement style can be broken into two components: the movement texture and the slowly-changing position relative to the person. We further note that these two components can be separated and handled independently to a degree, and that a particular movement detail (such as moving abruptly to show aggression) will have the same or very similar interaction impact over a range of positions relative to the human (e.g., immediately behind, 0.5m behind, slightly to the side, etc.).

If we consider these two path components as the high- and low-frequency components of the movement path, then it illustrates how we can generate output movements as follows: we can first solve for the low-frequency (relative to human position) component, and then superimpose the high frequency component (the movement texture) on this path.

The final challenge of this approach is to construct movement in real time interaction to a person's movements, without any pre-processing or path look-ahead available.

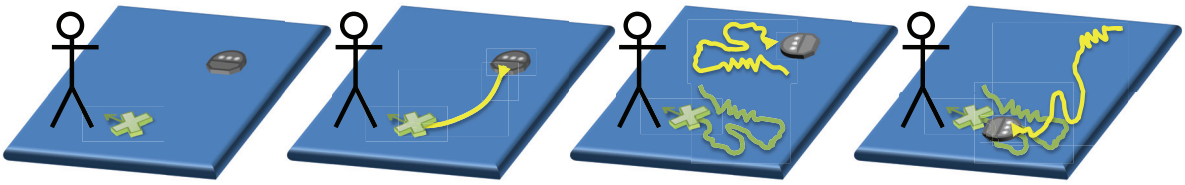
4.2 Motion Generation

We employed an inverse-kinematics model (basic trigonometry in this case) of our robot to generate both the high and low frequency components of movement; this further ensured that our calculated target motion is reproducible by our particular robot.

When provided with a target robot location from Puppet Master ($x, y,$ and θ with respect to the person), we use the model to solve for robot movement commands (*velocity* and *turning radius*) to move it to that target location (Figure 2a,b). This is the *low-frequency* component of the desired movement as it changes relatively slowly over time. However, the robot cannot instantly reach or keep up with the target as it moves (from ongoing generation), and the texture (or details) are lost as this method prioritizes reaching the target.

Separately, we calculate how the robot can reproduce the exact desired path (Figure 2c) by solving for the delta movements (changes in direction and location). This maintains the high-frequency component, but drift in the robot's imperfect movements means that the robot soon loses its low-frequency relative localization.

Finally, we combine the above two components using a weighted average of the resulting robot commands, the velocity and turning radius (Figure 2d). Focusing on the high-frequency component results in better texture retention but more location drift, and focusing on the low-frequency component has the opposite effect. We use a 70/30 high/low-frequency balance, selected through experimentation. The intuition is that existing high-frequency robot movements which tend to move away from the target location are slightly modified to correct their direction (to help meet the low-frequency goals), while movements that are already directed toward the overall target are generally unchanged. Although this solution is a compromise between



(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 2. The algorithm robot-movement frequency analysis principle.

relative position and texture, and neither is perfect, given our emphasis on motion style, this is perfectly acceptable as long as the desired result is communicated.

The overall process of how this fits with the Puppet Master algorithm is shown in Figure 3. During *demonstration*, input is fed directly into the Puppet Master base algorithm. For *generation*, the Puppet Master output is filtered through the kinematic model as a form of frequency analysis, and both outputs are weighted to produce the final robot command.

We believe that this solution is a unique method for filtering a desired movement against robot capabilities while focusing on the balance of texture and locomotion, and can be used for the general problem of applying unconstrained movements to the rigid constraints of real robots. We believe that this is not limited to locomotion path only but can be applied if a robot system must generate reactive movement in an unpredictable context without look-ahead capabilities. Our method can be used to generate motion while focusing on maintaining style if an inverse kinematics solver is available that enables the robot to: a) calculate the closest possible fit to reproducing that path and b) calculate the shortest-path route to an absolute configuration target.

5. Experienced-Programmer Design Critique

We present a design critique that addresses various aspects of how our user-centered SBD authoring approach differs from more-traditional programming methods. Other evaluation questions relating to this system are detailed in related work [Young et al., 2013, 2012].

We recruited experienced programmers to create interactive robot behaviors both programmatically and using our SBD broomstick interface. These participants were qualified enough to program interactive behaviors, giving them a unique comparison vantage point to provide an analysis of the trade-offs and benefits of in comparison to SBD.

We recruited four experienced programmers from our graduate-student lab (3 male, 4 female) who did not have prior exposure to our work. We asked the programmers to create the same set of stylized robot movement behaviors using: 1. Java programming via a provided API, and 2. our broomstick SBD tangible interface, where they could re-train as much as they wished (Figure 4). Participants were given two hours for the programming task, and no time limit was enforced for the SBD task. The behaviors were 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*).

Given our exploratory goals our aim was to elicit qualitative reflections on participant experiences and opinions: we see this as a precursor to direct more formal experiments. As such, we encouraged participants to verbally reflect on their interaction during the process and conducted a post-study semi-structured interview.

5.1 Results and Discussion

All participants took the full two hours to program their stylized 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, the participants took from 14 to 30 minutes (including reiteration and discussion), and noted that they found the SBD broomstick interaction much easier than programming. By using the demonstration system they "did not have to think technically or analytically," and could more-easily program movement styles. As such "there is a huge time-saving potential here." One reason cited for the broomstick's success is that people are very skilled at

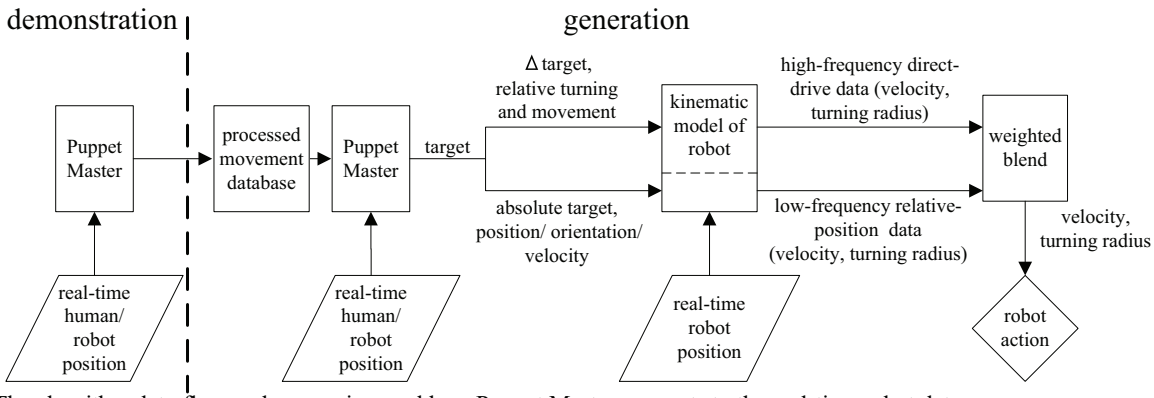
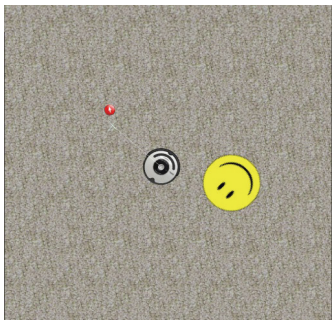


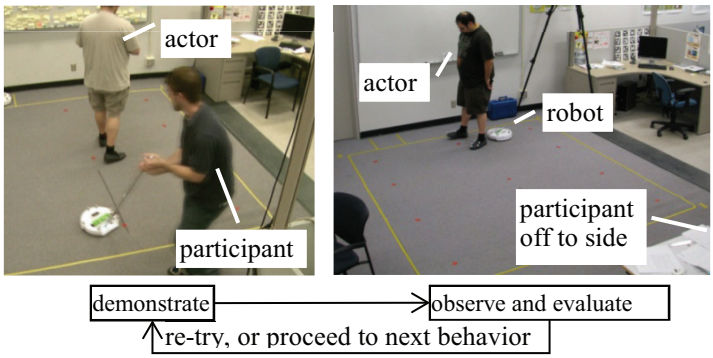
Figure 3. The algorithm data-flow and processing, and how Puppet Master connects to the real-time robot data.

programming condition



a) Programming test-bench.

style-by-demonstration condition



b) Demonstrate and observe cycle for broomstick SBD.

Figure 4. Participants were given two hours with the test bench to program the *polite*, *stalker*, *happy*, and *burglar* behaviors. Following, participants used the broomstick SBD interface to create the same behaviors, one at a time.

"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 SBD 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." Thus 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 interface and algorithm. We point out, however, that people learners 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.

All programmers were observed to act the characters using their entire bodies, making faces, laughing, etc. This latter point supports the idea that demonstrating style makes sense to people and involves their innate social and emotional interaction skills. We believe that the SBD benefits would be dramatically stronger for non-programmer users who are not otherwise able to create interactive behaviors.

When asked which they preferred, all 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 programmer "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 the programming condition. For example, general programming difficulties still exist: it's "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." Thus while programming does offer control, there is still a great deal of uncertainty.

Programmers mentioned that by focusing on style during programming 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 an animation smoothing approach), suggesting that the style elements may not be obvious unless emphasized. One noted that doing the programming before demonstration highlighted the sheer difficulty of the real-time problem, and helped them to appreciate the demonstration system.

One participant pointed out that the inherent inaccuracy of the demonstration system is not necessarily a problem, as perhaps the broomstick can be used to capture a rough behavior and serve as a medium fidelity prototyping method for behaviors that can be later programmed more thoroughly. Programmers also provided 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 variables to tweak when observing the result (e.g., as tangible knobs or a hand-held interface).

6 Conclusion and Future Work

In this paper we presented the idea of teaching robots *interactive, stylistic locomotion* by *demonstration*. We detailed our technical solution, introduced a novel robot-broomstick interface, and presented an evaluation that helped verify the applicability and accessibility of our approach in relation to traditional programming and provided insight into some of the tradeoffs between the approaches.

SBD provides a new way for people to communicate with and program robots. Looking forward, we intend to explore this point, and to investigate how SBD interaction and interfaces can be further improved to focus squarely on how people will want to teach and explain things to their robots.

References

[Amaya et al., 1996] Kenji Amaya, Armin Bruderlin, and Tom Calvert. Emotion from Motion. *Psychological Reports*, 18(12): 222–229. 1996 Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.6.2096&rep=rep1&type=pdf

[Dinerstein et al., 2006] Jonathan Dinerstein, Parris K. Egbert, and Dan Ventura. Learning Policies for Embodied Virtual Agents Through Demonstration. *Proceedings of the International Joint Conference on Artificial Intelligence IJCAI*, 6. 2006

[Frei et al., 2000] Phil Frei, Victor Su, Bakhtiar Mikhak, and Hiroshi Ishii. curlybot. *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '00* (pp. 129–136). New York, New York, USA: ACM Press. 2000

[Gallaher, 1992] Peggy E. Gallaher. Individual differences in nonverbal behavior: Dimensions of style. *Journal of Personality and Social Psychology*, 63(1): 133–145. 1992

[Gribovskaya and Billard, 2008] Elena Gribovskaya, and Aude Billard. Combining Dynamical Systems Control and Programming by Demonstration for Teaching Discrete Bimanual Coordination Tasks to a Humanoid Robot. *Human-Robot Interaction 2008, HRI '08*. 2008

[Heider and Simmel, 1944] Fritz Heider, and Marianne Simmel. An experimental study of apparent behavior. *The American Journal of Psychology*, 57(2): 243–259. 1944

[Igarashi and Igarashi, 2009] T. Igarashi, and Yuki Igarashi. Implementing as-rigid-as-possible shape manipulation and surface flattening. *Journal of Graphics, GPU, and Game Tools*, 14(1): 17–30. 2009

[Kanda et al., 2006] Takayuki Kanda, Masayuki Kamasima, Michita Imai, Tetsuo Ono, Daisuke Sakamoto, Hiroshi Ishiguro, and Yuichiro Anzai. A humanoid robot that pretends to listen to route guidance from a human. *Autonomous Robots*, 22(1): 87–100. 2006

[Lee and Lee, 2004] J. Lee, and K. H. Lee. Precomputing avatar behavior from human motion data. *Proceedings of the 2004 ACM SIGGRAPHEurographics symposium on Computer animation SCA 04*, 68(2): 79. 2004

[Maulsby et al., 1989] David L. Maulsby, Ian H. Witten, and Kenneth A. Kittlitz. Metamouse: specifying graphical procedures by example. *ACM SIGGRAPH Computer Graphics*, 23(3): 127–136. 1989

[Norman, 2002] Donald A. Norman. *The Design of Everyday Things*. *Human Factors and Ergonomics in Manufacturing* (Vol. 16, p. 272). Basic Books. 2002

[Raffle et al., 2004] Hayes Solos Raffle, Amanda J. Parkes, and Hiroshi Ishii. Topobo. *Proceedings of the 2004 conference on Human factors in computing systems - CHI '04* (pp. 647–654). New York, New York, USA: ACM Press. 2004

[Suay et al., 2012] Halit Bener Suay, Russell Toris, and Sonia Chernova. A Practical Comparison of Three Robot Learning from Demonstration Algorithm. *International Journal of Social Robotics*, 4(4): 319–330. 2012

[Thomas and Johnston, 1995] Frank Thomas, and Ollie Johnston. *The Illusion of Life: Disney Animation* (1 edition.). Disney Editions. 1995

[Tremoulet and Feldman, 2000] P. D. Tremoulet, and J. Feldman. Perception of animacy from the motion of a single object. *Perception*, 29(8): 943–951. 2000 Retrieved from http://www.perceptionweb.com/abstract.cgi?id=p3101

[Young et al., 2008a] James E. Young, Richard Hawkins, Ehud Sharlin, and Takeo Igarashi. Toward Acceptable Domestic Robots: Applying Insights from Social Psychology. *International Journal of Social Robotics*, 1(1): 95–108. 2008

[Young et al., 2008b] James E. Young, Takeo Igarashi, and Ehud Sharlin. Puppet Master : Designing Reactive Character Behavior by Demonstration. *Young*, 193(5): 183–191. 2008

[Young et al., 2013] James E. Young, Ehud Sharlin, and Takeo Igarashi. Teaching Robots Style: Designing and Evaluating Style-by-Demonstration for Interactive Robotic Locomotion. *Human-Computer Interaction*, in press. 2013

[Young et al., 2012] James Young, Kentaro Ishii, Takeo Igarashi, and Ehud Sharlin. Style by demonstration. *Proceedings of the 2012 ACM international conference on Intelligent User Interfaces - IUI '12* (p. 41). New York, New York, USA: ACM Press. 2012