Bodystorming Human-Robot Interactions

David Porfirio,[1] **Evan Fisher,**[2] **Allison Sauppé,**[2] **Aws Albarghouthi,**[1] **Bilge Mutlu**[1]
[1] University of Wisconsin–Madison, Madison, Wisconsin, USA
[2] University of Wisconsin–La Crosse, La Crosse, Wisconsin, USA
{dporfirio,aws,bilge}@cs.wisc.edu, {fisher.evan,asauppe}@uwlax.edu

ABSTRACT
Designing and implementing human-robot interactions requires numerous skills, from having a rich understanding of social interactions and the capacity to articulate their subtle requirements, to the ability to then program a social robot with the many facets of such a complex interaction. Although designers are best suited to develop and implement these interactions due to their inherent understanding of the context and its requirements, these skills are a barrier to enabling designers to rapidly explore and prototype ideas: it is impractical for designers to also be experts on social interaction behaviors, and the technical challenges associated with programming a social robot are prohibitive. In this work, we introduce *Synthé*, which allows designers to act out, or bodystorm, multiple demonstrations of an interaction. These demonstrations are automatically captured and translated into prototypes for the design team using program synthesis. We evaluate *Synthé* in multiple design sessions involving pairs of designers bodystorming interactions and observing the resulting models on a robot. We build on the findings from these sessions to improve the capabilities of *Synthé* and demonstrate the use of these capabilities in a second design session.

Author Keywords
Human-robot interaction; interaction design; ideation; brainstorming; design tools; program synthesis

INTRODUCTION
Robots that are designed to interact with people using human social norms of interaction present a complex design space and a new set of design challenges. Designers must not only create solutions that effectively address user needs and expectations, but these solutions must also closely follow acceptable interaction behavior, both to improve user satisfaction with the solution and to prevent breakdowns due to social-norm violations. For example, a hospital delivery robot that "beeps" as it navigates through a patient ward might effectively inform the hospital staff of its status, but it might also be seen as disrespectful by patients who are ill or in recovery [35]. Although

UIST'19, New Orleans, LA

ACM ISBN 123-4567-24-567/08/06…$15.00
DOI: http://dx.doi.org/10.475/123_4

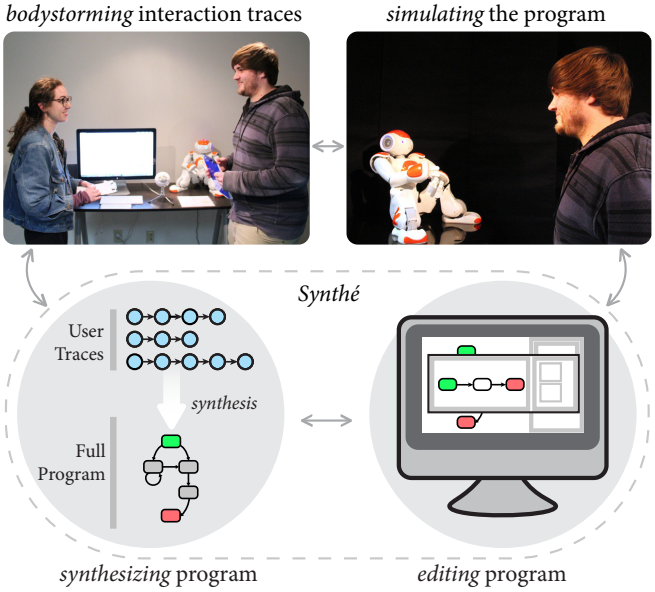

Figure 1. *Synthé* captures designers' demonstrations, synthesizes an interaction, and allows designers to edit and simulate the interaction.

there is a rich body of literature on human social behavior across the fields of social psychology and interpersonal communication, this literature may be unknown, inaccessible, or impractical to designers. How can designers more effectively draw on their intrinsic understanding of human interactions in creating human-robot interactions?

We propose to apply the design ideation method of *bodystorming* [55] to support interdisciplinary design teams with varying formal design experience explore and support complex human-robot interactions. Although these designers do not have a *formal* understanding of human interactions through their training or reading of the research literature, they instinctively navigate such interactions on a day-to-day basis. Our proposed method leverages this instinctive ability and involves members of design teams "acting out" or "role-playing" how they expect the human-robot interaction to flow. However, although such an approach might enable designers to devise human-robot interactions that are more closely aligned with human interactions, turning role-playing sessions into application prototypes still involves a significant amount of robotics implementation, hindering the practical use of such a method by designers who may not have the necessary background to

implement design ideas. To enable design teams to rapidly explore and prototype ideas, our proposed method uses a new *program synthesis* technique, specifically a form of automata learning [36], to automatically translate demonstrations into prototypes that the design team can evaluate on robots.

In this paper, we present our application of bodystorming within *Synthé*, a design environment that supports design teams in the creative exploration and prototyping of design ideas for human-robot interaction. Using this approach, designers with no expertise in robotics, programming, or social behavior can, within minutes, create interactive programs on a social robot. For example, a team designing a robot that assists travelers at an airport can perform how the interaction should play out across several demonstrations. Our system will then synthesize a program, semantically represented as a Mealy machine, from various constraints, including (1) a set of programming building blocks, specified beforehand, that represent designer actions recognizable by *Synthé*, and (2) multiple "traces" of social interaction demonstrated via the speech and gesture from the design team. At any point in the design process, designers can review and modify the input into the synthesis engine, test the prototype program on a robot, or provide additional traces until they are satisfied with their design. We evaluated our system in 12 design sessions involving pairs of designers; improved the design method and our system based on our findings from these sessions; and demonstrated the use of the updated method and system in another design session. The contributions of our work include:

- A *novel application* of bodystorming that enables design teams to rapidly explore and prototype human-robot interactions;
- A *software environment* that captures and synthesizes multiple designer demonstrations into prototype programs using program synthesis;
- An *empirical understanding* of how our proposed method and our system might support design teams in devising human-robot interaction solutions.

RELATED WORK

Embodied Design Methods

Using one's body in the interaction design process enables the designer to leverage the interconnectedness of the body and mind, incorporate human kinesthetics into the final product, and enhance collaboration with other designers [30]. *Bodystorming* is one such design technique in which designers brainstorm by situating their bodies in the context of the interaction being designed in order to gain insight into the user experience [15, 16, 47]. Bodystorming is achieved by role-playing an interaction with actors and props [14], which may or may not entail improvisation, or by immersing oneself in an environment similar to that being designed for [40]. Bodystorming has been applied to the design of mobile learning experiences [55] and robotic products [61], but no work has used bodystorming to synthesize human-robot interaction programs.

Based on bodystorming, *embodied sketching* involves improvisation in the design of experiences supported by technology rather than the technology itself [33]. Improvisation is also central to *embodied design improvisation*, a multi-stage generative and analytical design process that makes designers' tacit understanding of the design space explicit to other stakeholders in the design process [51, 52]. Prototypes are key in evaluating design ideas generated from this method, such as a robot ottoman that communicates primarily through movement [53]. Similarly, the *Personality* design method uses a series of improvisation and brainstorming steps to assign and implement behaviors based on human personality types [57].

Reflective Design Methods

Reflective design centers on the key insight that designers reflect on their experiences to shape their insights in the design process [11, 20] and has permeated a variety of other domains, such as city planning, education, and medicine [60]. HCI researchers have found numerous ways to incorporate reflective design practices as computers have entered contexts designers might otherwise be unfamiliar with [50], such as value-sensitive design [10, 22], participatory design [9, 34], and critical design [5, 23]. One particular area of interest in our work is the insight from Schön that designers "know more than they can say," being able to exhibit far more of their knowledge by demonstrating it in context than describing it to others out of context [48]. Using a combination of *knowing-in-action* and *reflection-in-action* to define a problem [49], designers can modify and expand their understanding of a problem space during its definition. With knowing-in-action, designers use their intuitive knowledge of the design space to act out and explore the actors, goals, and constraints. During this process, they constantly reflect on their actions and what is not yet fully articulated, improvising additional scenarios to more completely define the design space.

Design Tools & Environments

Those with the insight necessary to design an application for a particular context, such as end users, domain experts, and designers, may lack the technical background necessary realize their ideas. Visual programming environments (VPE's), such as *Code3*, aim to address this challenge by abstracting programming concepts into high-level graphical components to assist novice users [27]. The VPE's *Interaction Blocks* [46] and *RoVer* [42] abstract social robot behaviors into reusable building blocks, a technique we also use in *Synthé*. Alternatively, programming by demonstration (PbD) allows novice programmers to physically demonstrate the required sequence of actions on a robot that are then automatically captured and translated into a program [4, 7]. PbD algorithms are able to function in increasingly complex environments, such as recognizing the relative positioning of objects in the environment [38], breaking the task into discrete segments [38], recognizing partial-ordering of those segments [37], and understanding task constraints [41]. While PbD has been successfully applied to tasks in both in-home [1] and commercial [54] settings, the limitations of current PbD algorithms include their focus on tasks comprised of goals that require physical manipulation and the requirement to visually confirm completion.

Various technological advancements have increased the potential to program via human-body demonstrations, such as the recognition of full-body gestures [18] and the retargeting

of motion between figures with different body proportions [24]. Design tools that capture full-body movement include systems in which actors' movements are used to edit the motion of an animated character [21]; trainers' movements are authored and edited in order to teach novices expert motions, [2]; captured motion assists programmers in the generation, manipulation, and monitoring of visual input to and output from a camera-based program [29]; and non-expert illustrators are assisted by the automatic creation of motion illustrations based on captured sequences of movement [17]. The interface may also learn an optimal motion by demonstrating candidate motions to and obtaining ratings from an observer [31].

Computational Methods

Various computational approaches exist for the automatic construction of programs and automata from examples. The classic automata learning approach, L* [3], relies on an active loop of querying the user for examples and asking the user to confirm that the automaton it learns is the right one. Our setting is passive, in that we only have positive examples, given through bodystorming, and want to learn an automaton that encompasses them. Our work falls in the category of symbolic synthesis algorithms (see Solar-Lezama et al. [56] for the pioneering work and Gulwani et al. [26] for a survey), as it relies on a reduction of the problem to a combinatorial search using satisfiability (SAT) solvers and their first-order extensions (SMT solvers). Our work extends work by Neider [36] that encodes to the Mealy-machine setting, as we treat human and robot actions as inputs and outputs, respectively. Additionally, we consider an approximate learning setting, as we may not be able to find an automaton that precisely captures all observed interactions (e.g., due to conflicting demonstrations).

SYNTHÉ: DESIGN AND IMPLEMENTATION

In this section, we detail the design, algorithms, and implementation of *Synthé*, our software environment for facilitating bodystorming of human-robot interactions, using a running example. The workflow of *Synthé* is shown in Figure 1:

- *Bodystorming:* When designers *bodystorm* an interaction, they act out the scenario multiple times, resulting in a series of demonstrations that highlight different paths the scenario might follow. With *Synthé*, one designer plays the robot's part, while the other plays the human. Demonstrations are computationally represented as a set of *traces*, where each trace represents one demonstration of the human-robot interaction, captured as a sequence of alternating human and robot actions throughout the demonstration.
- *Program synthesis:* The traces resulting from bodystorming are fed into the program synthesis engine that generates a single program *generalizing* the observed demonstrations.
- *Visualization and editing: Synthé* enables viewing and modifying the program in a visual programming environment.
- *Simulation:* Finally, designers can compile and deploy the resulting program on a physical robot. After simulating the interaction, they may decide to modify it by continuing their bodystorming session and providing additional traces.

In the remainder of this section, we present the aforementioned components of *Synthé* in detail. Our running example entails the design of a delivery robot interaction, where the robot must deliver a package to a human. In this example, the human must verify their identity before receiving the package.

Bodystorming: Capturing Intent

The first and most crucial aspect of *Synthé* is capturing the bodystorming session of the design team. Since one designer acts as the human and the other as the robot, we view a demonstration through bodystorming as a *trace* of human and robot actions; formally, a trace t is a sequence of the form

$$\langle h_1, r_1 \rangle, \ldots, \langle h_n, r_n \rangle$$

where each pair $\langle h_i, r_i \rangle$ represents a human *input* action h_i and a robot *output* action r_i. For instance, the human action h_i may be saying "hello", while the robot reply r_i may be performing a more complex action such as saying "hello" *and* simultaneously performing a waving gesture.[1] We visualize traces as sequences of transitions and states, where transitions are annotated with both the human action (labeled **H**) and robot response (labeled **R**).

For example, Figure 3 (left) depicts two possible traces for the delivery robot. In the first trace, the human inputs a greeting and the robot asks for the human's identification. The human then attempts to receive the package, and the robot asks once more for the human's identification. Following the second query, the human denies that they have the appropriate identification, and the robot says farewell. In the second trace, after querying the human's identity, the human confirms their identity. The robot expresses gratitude, and the human motions to receive the package. The robot hands off the package to the human, and both bid each other farewell.

Throughout the bodystorming session, the design team demonstrates multiple interaction traces. The algorithmic core of *Synthé* is agnostic to the complexity of the action language, such that any combination of inputs can be treated as a robot output action r_i. In practice, *Synthé* captures (1) speech and (2) gestures; thus, each action is a combination of speech and gesture, either of which can be *null*, e.g., if the designer does not gesture. Gestures are not included in our running example.

Capturing Speech

Designers manually begin and end *Synthé*'s bodystorming recording. During the recording, designer audio is recorded and utterances are classified into one of several prespecified categories using natural language processing. The classification categories depend on the scenario, which in the running example of the delivery robot included *Greeting*, *QueryID*, *ConfirmID*, *DenyID*, *Gratitude*, *Receive*, *Handoff*, and *Farewell*.

Each designer is instrumented with a unidirectional lapel microphone that captures speech. We use off-the-shelf software to transcribe and classify timestamped speech. The timestamps determine the turn of talk between the human and the

[1] Our formulation assumes human actions are followed by robot actions. In cases, for example, where the robot initiates an interaction, we simply treat the human action as an empty (*null*) action.

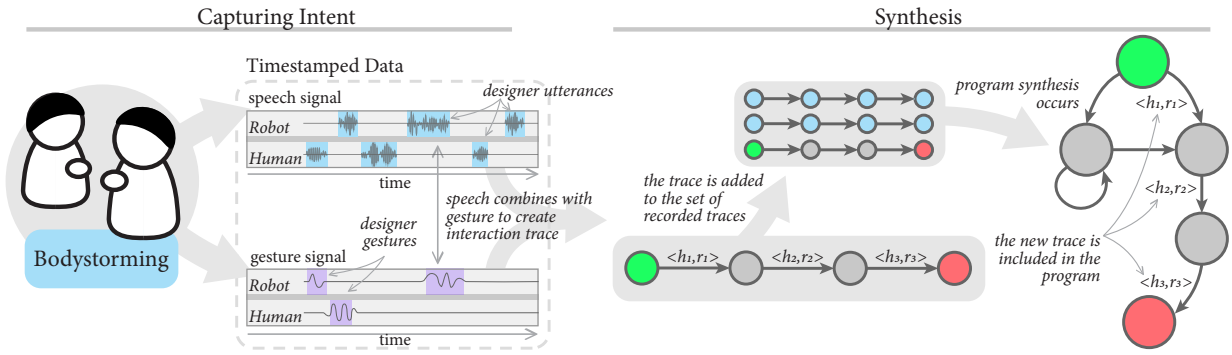

Figure 2. The pipeline from *Synthé* capturing designer speech and gesture signals, to converting the signals to a trace, to synthesizing an interaction based on all traces provided thus far. The green circles represent the start states of the interaction, while the red circles represent the end states.

robot, allowing *Synthé* to generate traces from demonstrations (Figure 2).

Gesture Recognition

Each designer is instrumented with a wireless armband that integrates a gyroscope and an accelerometer to capture arm motion and changes in orientation. The designer provides a single example of each gesture for *Synthé* to recognize, which is used by fast dynamic time warping (fastDTW) to classify any subsequent gestures [45]. Gestures are timestamped to couple the robot's gestures with its speech (Figure 2). *Synthé* can classify any performed gesture with reasonable accuracy as long as examples of the gesture are provided beforehand.

Program Synthesis

The bodystorming session results in a set of traces T representing possible trajectories of the intended interaction. The technical core of *Synthé* constructs a program that mimics the traces T while addressing two challenges: (1) ability to generalize the behavior demonstrated in the traces, instead of only capturing the finite behavior exhibited in the traces, and (2) robustness to conflicting behaviors in T, e.g., the robot performing a different action in each trace in response to the same human action. To address these challenges, we present a novel program synthesis algorithm that constructs Mealy automata representing the bodystormed interaction. Our algorithm constructs a set of constraints in *first-order logic* (FOL) whose solution is a *minimal* automaton that captures *most* transitions in the traces T. The minimality constraint serves as an *inductive bias* to generalize the set of given traces, e.g., by detecting iterative behavior and constructing loops. To illustrate this core idea, Figure 3, right, depicts the resulting automata derived from the traces in Figure 3, left. The interaction completely captures the transitions of both traces. As a Mealy automaton, the transitions are annotated with both human actions and robot responses. The synthesized program includes loops, inferred by our algorithm, including (1) the robot repeatedly querying the human for their ID if the human tries to receive before confirming their ID and (2) the robot handing off multiple objects.

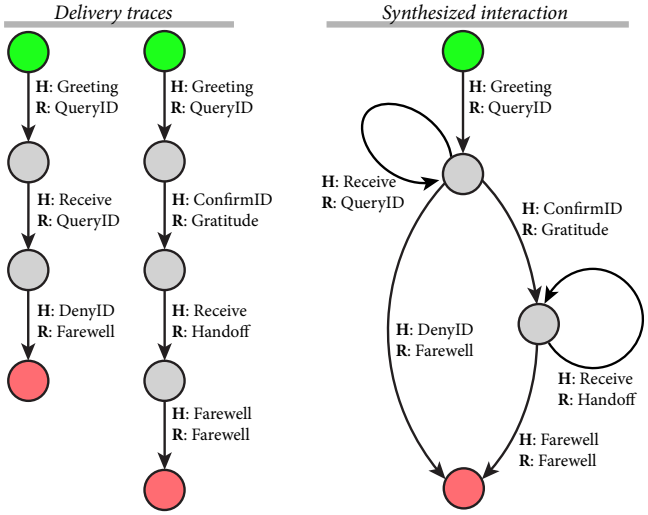

Figure 3. *Left:* Example traces representing possible interaction trajectories. *Right:* The resulting interaction synthesized from the examples.

Formalizing our Program Model

A program is represented as an automaton (a *Mealy* machine) with a finite set of *robot states* $S = \{s_1, \ldots, s_n\}$ and two *transition functions*:

$$f_A : S \times H \to R \qquad f_T : S \times H \to S$$

The *action transition* function f_A defines how the robot reacts to human actions: it maps human actions (set H) to robot actions (set R), depending on the current state (S). The *state transition* function f_T defines how the state of the robot changes depending on the observed human action.

We always treat state s_1 as the initial state in which the robot begins. An *execution* of the automaton is a sequence

$$s_1, \langle h_1, r_1 \rangle, s_2, \ldots, s_{m-1}, \langle h_{m-1}, r_{m-1} \rangle, s_m$$

Notice that automaton executions correspond directly (after removing the states s_i) to the traces captured by bodystorming. Our goal is to construct a small automaton whose executions capture most of the traces in a given set T.

Automata Learning Algorithm

The main challenge in learning an automaton from traces is the exponential number of automata with respect to the number of states and possible actions. Because naïve enumeration is impractical, we exploit symbolic methods to efficiently encode and search the space of automata. Specifically, we present

a new automata synthesis technique that encodes the space of automata in first-order logic and uses powerful *satisfiability modulo theories* (SMT) [6] solvers to discover solutions. Technically, our approach extends that of Neider [36] to the Mealy-machine setting and approximate learning of automata.

Our algorithm constructs the two transition functions defining an automaton, f_A and f_T. We will encode a set of first-order constraints whose solutions are functions f_A and f_T (formally, we use the first-order theory of *uninterpreted functions*).

Constraint on f_T—Our first constraint C_T ensures that, for each trace $t \in T$, the sequence of human actions are manifested in some execution of the automaton, namely in the state transition function f_T. Formally, for every trace $t = \langle h_1, r_1 \rangle, \ldots, \langle h_n, r_n \rangle$, we define the constraint

$$c_T^t \triangleq \exists x_1, \ldots, x_{n+1}. \left[x_1 = s_1 \wedge \bigwedge_{i=1}^{n} f_T(x_i, h_i) = x_{i+1} \right]$$

where the variables x_i range over the set of states S. Now, taking all the traces together, we define the overall constraint

$$C_T \triangleq \bigwedge_{t \in T} c_T^t$$

Constraint on f_A—Our second constraint, C_A, ensures that for each trace $t \in T$, the sequence of robot actions are manifested in some execution of the automaton, namely in the action transition function f_A. Formally, for every trace $t = \langle h_1, r_1 \rangle, \ldots, \langle h_n, r_n \rangle$, we define the constraint

$$c_A^t \triangleq \exists x_1, \ldots, x_n. \left[x_1 = s_1 \wedge \bigwedge_{i=1}^{n} \underbrace{f_A(x_i, h_i) = r_i}_{\text{soft constraint}} \right]$$

Now, for all traces, we define the overall constraint

$$C_A \triangleq \bigwedge_{t \in T} c_A^t$$

Constraint on initial and final states—Our final constraint, C_S, designates a specific state $s_f \in S$ as a final (termination) state, where (1) all traces lead to s_f and (2) s_f does not have any outgoing transitions. We omit the formal definition of C_S here.

Constraint Solving—Finally, now that we have defined the constraints, any solution (model) of the formula $C_T \wedge C_A \wedge C_S$ results in an automaton over robot states S—by defining the functions f_A and f_T—such that the automaton represents all traces T from the bodystorming session. However, because there may be traces in T that exhibit contradictory robot behavior, and we need to minimize the number of transitions in the automaton so as it does not exhibit arbitrary behavior, we pose this is as a *maximum satisfiability* problem, where instead of satisfying $C_T \wedge C_A \wedge C_S$ completely, we want to satisfy as many of the soft constraints as possible in C_A, which allow robot behavior to not exactly represent some traces in T. Next, we add a further soft constraint to minimize the number of transitions between states (by maximizing the number of transitions to the *sink* state s_f). The resulting combinatorial optimization problem can be solved using an off-the-shelf MaxSMT solver.

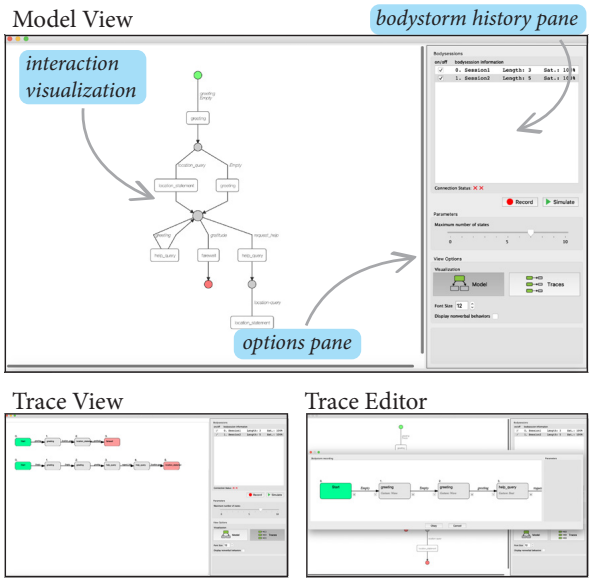

Figure 4. *Synthé*'s user interface, featuring the model view (top), trace view (bottom-left) and trace editor (bottom-right)

Supporting Visualization and Editing

Synthé includes a user interface, shown in Figure 4, that provides visual feedback on design progress and limited trace editing capabilities. The visualization pane allows designers to switch between viewing their in-progress program as a transition system (the "model view") and the set of traces provided thus far (the "trace view"). The history pane allows designers to view a summary of traces provided thus far, including the inclusion percentage for each trace. Users can click on a trace to open up the trace editor, which allows designers to view traces in more detail and edit them. Editing capabilities include modifying the speech-to-text, action classifications, and robot gestures corresponding to an action state. Underneath the history pane, users have the option to record a new trace or simulate the synthesized program, change the allowed size of the synthesized program, and change visualizations.

System Implementation

Synthé is primarily implemented in Python version 3.6.[2] The system connects to IBM Watson's Speech-to-Text service to convert speech to timestamped text [28], followed by Rasa's Intent Classification service trained using examples from a text classification dataset generator to classify each utterance [8, 44]. Myo armbands collect the acceleration and orientation data, which is fed to the gesture classifier [59]. *Synthé* then solves the resulting synthesis problem using the Z3 SMT solver [19]. After designers act out a demonstration and *Synthé* parses the utterances, synthesis of a new interaction occurs in the background while all other functionality of *Synthé* remains available to users. Designers may act out another demonstration, edit previously-acted traces, or simulate the most recently synthesized program, all while a new program is synthesized. Support for visualizing interactions is implemented in D3 [12].

We implemented the ability to simulate interactions in Python version 2.7 on a Softbank Robotics Nao robot, using version

[2]The source code is hosted at https://github.com/Wisc-HCI/*Synthé*.

2.1.4 of NaoQi [43]. The robot traverses the interaction as a Mealy machine, waiting for human input to determine the appropriate transition. We implemented the robot to only recognize speech input. Speech intents and referential entities are extracted using the same recognition and classification system used to capture designer intent. Speech input that cannot be accepted by the current state invokes a recovery transition; the robot states that it did not understand the human's speech, and the transition returns to the previous state to allow the human to repeat or provide clarification. For all robot responses within the synthesized interaction, the robot's speech is derived from the text extracted from the traces. The robot also chooses gesture behaviors based on the gesture classification associated with the current state in the synthesized interaction.

EVALUATION

Our evaluation tested *Synthé* with 12 design teams to obtain user feedback on bodystorming as a design method for human-robot interaction. We also tested *Synthé*'s computational performance. While the version of *Synthé* used in the performance evaluation is current, the version used in our user study did not support automatic gesture recognition or derive robot speech behaviors from designer demonstrations. Rather, robot speech was prespecified for each robot response. The feedback from our evaluation informed the improvements made to *Synthé*.

Design Sessions

Design Procedure—Figure 5 shows the study setup. Upon obtaining informed consent from both participants, the experimenter played a video demonstrating the bodystorming design method, and the capabilities of *Synthé*. Subsequently, the experimenter guided the pair in a brief bodystorming demonstration, explaining how to interpret and edit the demonstration in *Synthé* and how to simulate the resulting program on the robot. The pair was then informed about the two experimental scenario that they would be assigned: (1) the robot delivers a package to a human, or (2) the robot provides information to shoppers at a store about the locations and prices of items for sale. The participant pairs were given 20–25 minutes to create their designs and were not allowed to brainstorm beforehand.

After the design session, participants completed questionnaires that assessed the value of bodystorming as a design method and the usability of *Synthé*. The experimenter then initiated a

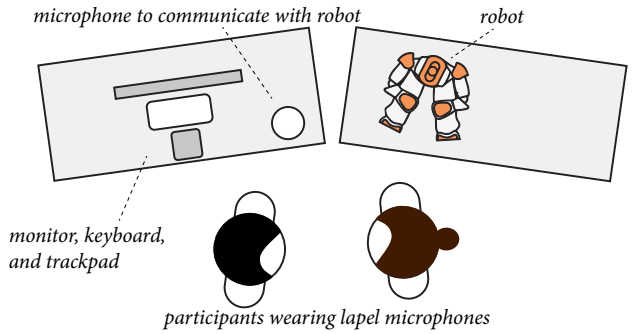

Figure 5. The physical layout of our evaluation. In this evaluation, participants did not wear armbands to capture gestures.

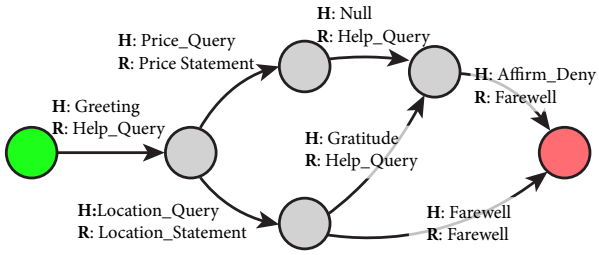

Figure 6. Example human-robot interaction design of a help desk robot created during the empirical evaluation.

retrospective think-aloud (RTA) [39, 25] by first demonstrating the RTA protocol to the participants and then asking them to engage in an RTA while watching a video recording of their design session. The video recordings consisted of the movement and speech made by participants synchronized with screen recordings of *Synthé* being used. After completing the RTA, the experimenter engaged the pair in a 10-minute semi-structured interview on their general experience.

Measurement & Analysis—Our evaluation primarily followed a qualitative approach to assessing how bodystorming, and *Synthé*, might support design teams in creative exploration and prototyping of human-robot interactions. The qualitative data included transcriptions of participant speech during the RTA and semi-structured interview. We also measured overall usability and user experience with *Synthé* using the USE questionnaire to assess the value of bodystorming as a design method [32] and the SUS questionnaire to assess the usability of *Synthé* [13]. We included an additional scale that assessed designers' perceived interaction design quality. Lastly, we measured system performance using timing data that describes how long *Synthé* takes to synthesize an interaction.

The qualitative data was analyzed following a modified *content analysis* process [58], which involved extracting an exhaustive set of *in vivo* codes, categorizing these codes to create a shared code library, clustering the codes in the library to identify general themes that emerged from the data, and identifying illustrative instances of data to characterize each theme.

Participants—We recruited 24 participants (8 males, 16 females), aged 18–27 ($M = 21.0$, $SD = 2.61$), who made up 12 pairs (2 male-male, 4 male-female, 6 female-female pairs) and represented our intended users to carry out design sessions. Participants came from a wide range of backgrounds, including design, engineering, humanities, and domain sciences such as education and health. Five participants had self-reported prior design experience. Participants from 10 pairs were personally acquainted with their partner.

Results

In this section, we present the five key themes that emerged from our analysis of the RTA sessions and interviews, the results from the design strategy analysis and the post-session questionnaires, and the performance evaluation of *Synthé*. Figure 6 shows an example of a design team's completed interaction synthesized from three example demonstrations.

Theme 1: Idea Generation and Feedback

We found that many designers generated ideas from sources external to the bodystorming process, such as remembering past experiences similar to the design scenario. For instance, P3 drew on experience from working in a store to design the help desk robot, stating that *"I used to work at a store so I think I was like trying to like, think back to like different scenarios I experienced, while working there."* Designers also relied on their understanding of social situations to predict how a robot might behave differently from a person:

> *I was kind of thinking it would be different too if a robot came to your door versus like a human. I don't know I just feel like it would be different interactions... So I feel like it would almost have to be more questions being asked too. I don't know just to make sure everything is right. (P24)*

Other participants generated ideas from the study materials given to them, namely the details about the scenario being designed for, such as the locations and prices of items in the store, or the list of possible speech classifications:

> *Um, what helped me was the speech classification keys, mm thing. 'Cause that way I could be like, what am I- what are the different ways I could classify speech, meaning like, these are the ways you could classify speech so somebody could've said this. (P14)*

In contrast to ideation, feedback on ideas and in-progress designs often arose from within the design process itself. In the demonstration stage of the workflow, one design team expressed that bodystorming caused them to think about the delivery interaction scenario in a different way:

> *Yeah, I mean, I. . . I don't, um, physically interact with a person most times I get a package of course so, uh, you kind of have to consider like, I see the [delivery] man, sure, but I never, you know, talk to him but it's. . . It was interesting to kind of like think about how don't actually experience this. (P11)*

Designers also received feedback from the visualization of the full interaction. In one case, P22 mentioned how the updated visualization after each demonstration guided the design team, saying *"As the thing went on we saw the thing build on the thing so then we like did our demonstrations based on how we wanted it to build."* Feedback often came from within the design team itself, where partners might disagree with each other's ideas, or guide the team's next steps:

> *I was kind of surprised, at first, when you kept saying, do it in the other order. I get you were trying to get it- the robot to recognize it, but I didn't think it would help. But then like, I was like well, we have no other choice, like, let's try to do the questions in the other order. (P7)*

Many designers used mistakes made during acting, or unexpected behavior displayed by the robot, to guide their design.:

> *Basically, whenever we knew that we messed up on something or it didn't go how we wanted it to we just thought of what we could do better next time, so even with the simulation when [the robot] didn't talk back to us we knew that we had to focus on that area when we were editing. (P23)*

Implications—This theme reinforces the main premise of *Synthé*, indicating that designers can indeed rely on their instinctive understanding of social interactions, including their prior experiences, to develop design ideas for human-robot interaction. Engaging in bodystorming is also a reflective process that enables designers to consider the nuances of what behaviors might be appropriate for a given scenario as well as how a robot might act differently from a person in a given scenario.

Theme 2: Constraints in Design Ideation

Many design teams introduced explicit *constraints* on their demonstrations, such as by creating detailed plans of the structure and content of future demonstrations. For instance, some design teams created interaction scripts, containing the specific speech utterances that each designer would say during their next demonstration, e.g., P11 recalling *"And then, uh, we wrote out all of our lines and stuff."* Although not necessarily planning out exact speech, some teams still carefully planned the flow of their demonstrations, while others rehearsed their demonstrations, before recording them:

> *Just, rather than just going to do it and be like, oh, can you do that again? We wanted to get a good cut right off the bat. And then go into detail on the program like we did. (P2)*

Either while acting out a demonstration or editing traces generated post-demonstration, many designers constrained their speech to adhere to what *Synthé* was expected to understand or what a robot was expected to understand in the real-world:

> *Yeah, and I think when we were talking, we were talking a lot more clearly and slower than I normally would talk to someone, just to make sure that the microphone could get what we were saying, or... (P4)*

Similarly, designers such as P24 constrained their interaction flow to avoid confusing the robot, stating *"So we had to do it so that it was all back and forth instead of one person saying two things right after another."* Some design teams also embodied static design roles, either by repeatedly acting in the same role during demonstrations, or by designating a single designer to use the graphical interface. For example, P16 stated that *"I was always the robot. Because we wanted to keep it consistent."*

While many teams followed a preset interaction structure and speech, some designers took a more fluid approach, switching their acting roles between demonstrations such as P17 and P18 or even improvising during their demonstrations:

> *We basically came up with uh, which scenario to go with, and we really just then improv'd the words as we were going, you know? (P22)*

Implications—These findings underline the need to support design teams in their planning process, such as in specifying actions that should or should not be recognized, which can also improve *Synthé*'s recognition and synthesis performance.

Theme 3: Mental Models of Synthé

Design ideation was affected by the mental models that designers built of *Synthé*, such as its ability to capture and synthesize the exact text that designers encoded into their demonstrations, e.g., P12 expected that the speech *"was just going to go from the computer to the robot."* Designers also likened the use of *Synthé* to "teaching" the robot how to interact:

It wait almost felt like teaching a little kid, in the sense of trying to kind of, like, explain to them how something works, or what is this, in that they don't understand, and you have to adjust what you're saying depending on what they're out- understanding is of what's going on. (P9)

Additionally, designers had expectations about how the simulations of their designs would play out:

Umm I guess... we wanted it to um... we were expecting it to say exactly what we said in the skit, but it didn't... it said something else. It was still like the right dialogue I guess, but it wasn't word for word what we were saying. (P20)

Many designers indicated that the robot exhibited behavior that they did not expect during simulation. For instance, P23 stated that *"It was picking up everything, just like it didn't seem like it was in the order we talked about it."* This finding highlights the key role that simulation plays in the design process by exposing design flaws and unexpected behavior in teams' designs. Designers also expressed dissatisfaction with the course of their simulations:

So I think we did this one again and we did it good and then we simulated it and it was kind of weird... Like it didn't work very well. (P22)

Despite unmet expectations and dissatisfaction with simulation that these designers expressed, some designers thought that simulation was a good idea in general, and many designers expressed a desire for more simulation in their workflows:

Yeah and simulating with the actual robot was cool too, because then you can like test it out. (P24)

And I think if we went back and like simulated it with the robot, it would have figured out. (P11)

Implications—Our findings indicate that designers' mental models of how *Synthé* worked affected their ideation and that helping build correct mental models is critical for *Synthé* to serve as an effective design tool. They also highlight the importance of prototyping and testing during the ideation process and thus the iterative nature of design ideation.

Theme 4: Acting Experience

Designers had mixed responses regarding their acting experience and whether they thought that acting was useful to the design process. Some designers had a positive experience with acting, finding it to assist with the design process and to add naturalness to their demonstrations:

Um but doing it face to face with the person I think helped us plan out our stuff a lot better. (P15)

But as long as we had the right speech it was fine but acting it out helped be like a real conversation. Because it seemed like more natural instead of just like scripted. (P23)

Others felt that bodystorming felt artificial:

It felt a little unnatural, even though, I guess we were trying to make it seem like the conversation was like a normal everyday kind of thing. (P4)

Implications—Our findings indicate that designers will have different levels of comfort with "acting out" design ideas and that these individual differences and preferences must be considered when adopting design methods such as bodystorming.

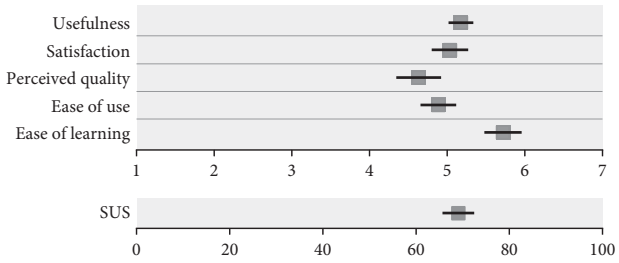

Figure 7. Results from the SUS and USE questionnaires and the measure of perceived quality. Error bars represent standard error.

Theme 5: Use of Gestures and Props

Some designers incorporated gestures into their acting to enhance their designs, and some felt that gesturing during their demonstrations made acting feel more natural. For instance, P8 expressed that *"it makes it like, more awkward to just like, "bye," and then like, stand there, like that just doesn't feel natural."* Designers even expressed the need for a more comprehensive library of gestures that could be encoded on the robot, e.g., P4 expressed that *"some of them, we kinda found you couldn't really use with gesture that was listed, or..."* Similarly, designers often incorporated props into their demonstrations. Multiple design teams thought that props added realism to their demonstrations:

Yeah, the props made it more realistic I think, because I don't think that I would've said like "here you go" if I wasn't holding anything in my hand. (P16)

Conversely, other design teams did not use gestures or props in order to focus on their speech:

I think we like more focused on...what we were going to say than like gestures or doing it or anything like that (P12)

Implications—Findings in this theme indicate a strong need to incorporate aspects of bodystorming beyond speech, including the ability to capture and represent the use of gestures and physical props, in *Synthé*. While some designers found these facets of bodystorming to be essential, others chose to focus on speech. Therefore, gestures and props must be incorporated in a way that does not become a barrier for designers.

Usability

Figure 7 shows data from the SUS and USE questionnaires. Designers' average SUS score was 69.1 ($SD = 16.7$). Within USE, their average score was 5.18 ($SD = 0.78$) for usefulness, 4.89 ($SD = 1.13$) for ease of use, 5.72 ($SD = 1.18$) for ease of learning, 5.04 ($SD = 1.15$) for satisfaction, and 4.63 ($SD = 1.41$) for the perceived quality of their own interaction designs.

Design Stages

Pairs spend 50% ($SD = 3.0\%$) of non-idle time brainstorming, 32% ($SD = 3.8\%$) editing, 9.1% ($SD = 1.3\%$) acting, and 8.9% ($SD = 2.9\%$) simulating (Figure 7.a). We define idle time as gaps between brainstorming, editing, simulating, or acting, comprising on average 0.92% ($SD = 0.01$) of total

design time, such as when pairs waited in silence for *Synthé* to analyze their speech. We define acting as blocks of time when *Synthé* records demonstrations, simulation as when the simulator is active, editing as when participants are in the trace editor and make changes to a trace, and brainstorming as the gaps between acting, editing, and simulating in which discussion occurs between participants.

Figure 7.b shows the average workflow for all 12 pairs. Each arc represents a transition from one phase to another phase, and its width represents the average frequency of that transition. We excluded a total of three brainstorming sessions from our workflow analysis due to the sessions being shorter than the shortest observed non-brainstorming session over all design teams (seven seconds). For example, the sequence *simulating→brainstorming→editing* would be treated as *simulating→editing* if the brainstorming session took less than seven seconds. We observed transitions from brainstorming to acting ($M = 24\%, SD = 5.2\%$) to be most frequent.

System Performance

We evaluated the performance of *Synthé* by measuring the computation time for synthesizing a program with five states for various amounts and lengths of randomly-generated traces provided to the system. Length is computed by the number of states in a trace, minus the start state. Figure 7.c shows the result of our analysis on a 2.9 GHz Intel Core i5 chip, testing each combination of trace number and length 10 times. We observed the average computation time for traces of length less than or equal to five to be consistently below one minute. The longest observed average time was observed to be 211 seconds ($SD = 6.27$) for eight traces of length nine. Completion time for lengths seven, eight, and nine begins to level off after six input traces, as *Synthé* returns, by design, the best possible solution once three minutes of finding an initial solution has surpassed. Our evaluation of performance reflects realistic prototyping with *Synthé*, as pairs on average supplied 3.25 traces ($SD = 1.48$) of length 4.72 ($SD = 1.82$).

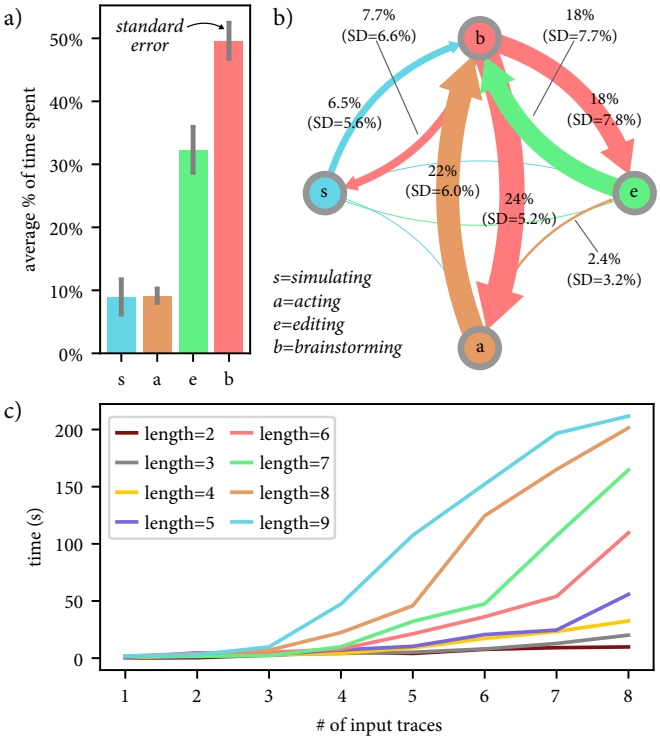

Figure 8. The average (a) percent of time spent in each design phase, (b) frequencies of transitions between phases, and (c) performance of *Synthé* as the number and length of input traces increases.

DESIGN IMPROVEMENTS

Gesturing—We added automatic gesture recognition to *Synthé* based on designers' use of gestures in our evaluation. Our solution also addresses designers' desire for more gestures to assign to the robot. Designers can now prespecify the types of gestures that *Synthé* should recognize, which will cause *Synthé* to ask for demonstrations of these gestures upon being started up. However, simulating new gestures on the robot requires programming the appropriate robot movements.

Robot Speech—To address the confusion that robot speech did not match participants' acted speech, in our updated implementation, the robot matched the acted speech by matching the current simulation state to a demonstration trace and choosing speech from the corresponding state in the trace.

Follow-up Study

Study Details—We conducted an informal extended design session of the updated *Synthé* with one pair of design students. Both participants were female and 21 years old. The procedure differed from the previous design study, such that (1) participants had up to one hour for their design session; (2) our RTA covered only the beginning and end of the session; and (3) the RTA focused on the updates made to *Synthé*.

Results—Although satisfied with the selection of gestures available to assign to the robot, designers did not purposefully employ physical acting with gestures or props for the duration of the session. One designer stated that gesturing was not a priority and that they do not convey as much as speech. During the interview, designers expressed some interest in using pointing gestures, although they highlighted that they would not have anything to point to given the lack of props.

Despite the speech updates made to *Synthé*, designers expressed dissatisfaction with the speech that the robot uttered. One designer described the difference in speech as a "broken connection." The second designer expressed that the most difficult aspect of the design "was getting the robot to reply the way that we wanted to." Despite these difficulties, designers described themselves as "impressed." The SUS scores for *Synthé* were 62.5 and 72.5 for each designer. Designers' USE scores for the design method were 4.13 and 5.63 for usefulness, 3.45 and 5.73 for ease of use, 3.25 and 7.0 for ease of learning, 3.43 and 6.43 for satisfaction, and 4.2 and 5.0 for perceived quality of the resulting interaction design.

DISCUSSION

Our evaluation and follow-up demonstration provide insights into the benefits and limitations of our design approach and the efficacy of *Synthé*, which we discuss below.

Implications from Empirical Evaluation

Our evaluation helps us understand the mechanisms behind idea generation, receiving feedback, strategies for planning and executing design ideas, and designers' understanding of the robot's capabilities and actions. Ideas generated arose from a wide range of sources, including prior experience, feedback from their partner, the robot, or from acting. Often such feedback was received positively and served to guide designers' design strategies, but could have served as a source for misleading feedback. For instance, in the first version of *Synthé*, designers were prevented from customizing the robot's speech behaviors, yet continued to strive to achieve this ability.

Additionally, enhancing the planning process may further encourage designers to generate ideas from within the interaction design process, as well as relying on external sources. Not only should features be included that support the heavy planning of demonstrations or the addition of constraints that allow *Synthé* to ignore certain undesirable actions, but features should be included that allow for designers to recover from when their demonstrations do not go according to plan. The ability to begin a demonstration from the middle of a trace, rather than from the beginning, could support the ability to recover. Similarly, the ability to begin a simulation from any state would allow designers to rapidly test quick fixes their design.

The results from our quantitative analysis show that true to bodystorming, brainstorming is an essential component of *Synthé*. We found that the majority of design time was spent brainstorming and that brainstorming was central to participants' workflows, as illustrated in Figure 8. The participant workflows also show that although less overall time was spent acting due to acting sessions generally being short, acting was performed frequently. We therefore conclude that *Synthé* supports bodystorming through brainstorming and acting.

Updated Version of Synthé

With the current version of *Synthé*, designers still expressed frustration when the robot's speech did not adhere to their exact expectations, even when adhering to such expectations would potentially cause the robot to contradict itself (e.g. the robot may state a different price or location for an item than what exists in its database). Future versions of *Synthé* may need to be more explicit in the capabilities of each component of the design process. Furthermore, although designers who used the updated version of *Synthé* were satisfied with the scope of available gestures, having a library of prespecified gestures may not be sufficient in certain design contexts. At an airport, for instance, the robot may need to employ a much wider variety of physical behaviors to suit the diverse backgrounds of the humans interacting with it.

Rapid Design and Prototyping

In addition to designers' feedback on our design method, we also demonstrated the ability to rapidly prototype human-robot interactions using our method in twenty-five minutes or less. While many designers expressed that their designs were not yet complete when their design time ended, designers objectively demonstrated the ability to create functioning prototypes within the allotted time. Additionally, although many designers described their design process as thoughtfully planned out either through outlining or rehearsing their demonstrations beforehand, we believe that *Synthé* accommodates this planned design strategy by requiring only a few traces before an interaction is synthesized within a brief amount of time and ready to be tested. In effect, designers have more time to plan their designs, since fewer demonstrations are required.

Limitations

In addition to the areas of improvement outlined above, *Synthé*'s capabilities are limited to high-level prototyping due lack of flexibility, namely the inability to manipulate low-level controls such as the timing and concurrency of robot behaviors. Future versions of *Synthé* must integrate these details into the trace editor or capture them from designers, as these low-level details are integral to programming successful human-robot interactions and common to other programming environments. Furthermore, *Synthé* does not support designers creating their speech and gesture categories from scratch, requiring them to prespecify interaction components, which also limits the ability to improvise interactions. Future versions of *Synthé* can support on-the-fly designer creation and modification of constraints, such as speech and gesture classifications, for the design scenario at hand. Future work could also enable the mimicry of designer gestures on the robot. Current capture and retargeting methods may not ensure effective mimicry for all robot platforms due to differences in geometry and kinematics.

Additionally, our method of interaction synthesis does not currently distinguish between different behaviors that the human or robot may emit within a single behavior class. Using the help desk robot as an example, the existence of a product cannot easily be accounted for within when answering an end user's query for the location of an item in the store. If designers want the robot to be able to make this distinction, they need to perform separate demonstrations for each store item, which assumes that speech classifications for each item exists. Future versions of *Synthé* need to address this issue, e.g., using a parameterized (symbolic) automaton alphabet.

CONCLUSION

In this paper, we presented *Synthé*, which enables pairs of designers to rapidly prototype human-robot interactions via bodystorming. *Synthé* uses a new symbolic program synthesis algorithm to generalize demonstrations into a program that can be deployed and simulated on a robot. We evaluated *Synthé* by asking designers to bodystorm different interaction scenarios, improved our system based on our findings, and demonstrated the improved system in a follow-up study. Our application of bodystorming within *Synthé* serves as an example for design-support tools to address the unique design challenges involved in creating interactive robotic technologies.

ACKNOWLEDGEMENTS

This work was supported by the National Science Foundation (NSF) award 1651129 and an NSF Graduate Research Fellowship. We thank Loris D'Antoni for pointing us to Neider's algorithm, and Linda Wu for her help in analyzing data.

REFERENCES

[1] Sonya Alexandrova, Maya Cakmak, Kaijen Hsiao, and Leila Takayama. 2014. Robot programming by demonstration with interactive action visualizations.. In *Robotics: science and systems*. Citeseer.

[2] Fraser Anderson, Tovi Grossman, Justin Matejka, and George Fitzmaurice. 2013. YouMove: enhancing movement training with an augmented reality mirror. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*. ACM, 311–320.

[3] Dana Angluin. 1987. Learning regular sets from queries and counterexamples. *Information and computation* 75, 2 (1987), 87–106.

[4] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. 2009. A survey of robot learning from demonstration. *Robotics and autonomous systems* 57, 5 (2009), 469–483.

[5] Shaowen Bardzell, Jeffrey Bardzell, Jodi Forlizzi, John Zimmerman, and John Antanitis. 2012. Critical design and critical theory: the challenge of designing for provocation. In *Proceedings of the Designing Interactive Systems Conference*. ACM, 288–297.

[6] Clark Barrett and Cesare Tinelli. 2018. Satisfiability modulo theories. In *Handbook of Model Checking*. Springer, 305–343.

[7] Aude Billard, Sylvain Calinon, Ruediger Dillmann, and Stefan Schaal. 2008. Robot programming by demonstration. *Springer handbook of robotics* (2008), 1371–1394.

[8] Tom Bocklisch, Joey Faulkner, Nick Pawlowski, and Alan Nichol. 2017. Rasa: Open source language understanding and dialogue management. *arXiv preprint arXiv:1712.05181* (2017).

[9] Susanne Bødker. 1991. Through the interface-A human activity approach to user interface design. *DAIMI Report Series* 224 (1991).

[10] Alan Borning and Michael Muller. 2012. Next steps for value sensitive design. In *Proceedings of the SIGCHI conference on human factors in computing systems*. ACM, 1125–1134.

[11] Terry Borton. 1969. Reach, Touch, and Teach. *Saturday Rev* (1969).

[12] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. 2011. D^3 data-driven documents. *IEEE Transactions on Visualization & Computer Graphics* 12 (2011), 2301–2309.

[13] John Brooke and others. 1996. SUS-A quick and dirty usability scale. *Usability evaluation in industry* 189, 194 (1996), 4–7.

[14] Marion Buchenau and Jane Fulton Suri. 2000. Experience prototyping. In *Proceedings of the 3rd conference on Designing interactive systems: processes, practices, methods, and techniques*. ACM, 424–433.

[15] Colin Burns, Eric Dishman, Bonnie Johnson, and Bill Verplank. 1995. Informance": Min (d) ing future contexts for scenariobased interaction design. In *BayCHI (Palo Alto, August 1995). Abstract available at http://www. baychi. org/meetings/archive/0895. html*.

[16] Colin Burns, Eric Dishman, William Verplank, and Bud Lassiter. 1994. Actors, Hairdos & Videotape–Informance Design. In *Conference Companion on Human Factors in Computing Systems (CHI '94)*. ACM, New York, NY, USA, 119–120. DOI: http://dx.doi.org/10.1145/259963.260102

[17] Pei-Yu Peggy Chi, Daniel Vogel, Mira Dontcheva, Wilmot Li, and Björn Hartmann. 2016. Authoring illustrations of human movements by iterative physical demonstration. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. ACM, 809–820.

[18] Gabe Cohn, Daniel Morris, Shwetak Patel, and Desney Tan. 2012. Humantenna: using the body as an antenna for real-time whole-body interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1901–1910.

[19] Leonardo De Moura and Nikolaj Bjørner. 2008. Z3: An efficient SMT solver. In *International conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 337–340.

[20] J Dewey. 1933. How We Think. A Restatement of the Relation of Reflective Thinking to the Educative Process, Boston etc.(DC Heath and Company) 1933. (1933).

[21] Mira Dontcheva, Gary Yngve, and Zoran Popović. 2003. Layered acting for character animation. In *ACM Transactions on Graphics (TOG)*, Vol. 22. ACM, 409–416.

[22] Batya Friedman, Peter H Kahn, and Alan Borning. 2008. Value sensitive design and information systems. *The handbook of information and computer ethics* (2008), 69–101.

[23] Bill Gaver and Heather Martin. 2000. Alternatives: exploring information appliances through conceptual design proposals. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. ACM, 209–216.

[24] Michael Gleicher. 1998. Retargetting motion to new characters. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*. ACM, 33–42.

[25] Zhiwei Guan, Shirley Lee, Elisabeth Cuddihy, and Judith Ramey. 2006. The validity of the stimulated retrospective think-aloud method as measured by eye tracking. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*. ACM, 1253–1262.

[26] Sumit Gulwani, Oleksandr Polozov, Rishabh Singh, and others. 2017. Program synthesis. *Foundations and Trends® in Programming Languages* 4, 1-2 (2017), 1–119.

[27] Justin Huang and Maya Cakmak. 2017. Code3: A system for end-to-end programming of mobile manipulator robots for novices and experts. In *2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI*. IEEE, 453–462.

[28] IBM. n.d. Speech to Text. (n.d.). `https://www.ibm.com/watson/services/speech-to-text/`, Accessed: 2019-07-15.

[29] Jun Kato, Sean McDirmid, and Xiang Cao. 2012. DejaVu: integrated support for developing interactive camera-based programs. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*. ACM, 189–196.

[30] Scott R Klemmer, Björn Hartmann, and Leila Takayama. 2006. How bodies matter: five themes for interaction design. In *Proceedings of the 6th conference on Designing Interactive systems*. ACM, 140–149.

[31] Heather Knight and Reid Simmons. 2017. An intelligent design interface for dancers to teach robots. In *2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 1344–1350.

[32] Arnold M Lund. 2001. Measuring usability with the use questionnaire12. *Usability interface* 8, 2 (2001), 3–6.

[33] Elena Márquez Segura, Laia Turmo Vidal, and Asreen Rostami. 2016. Bodystorming for movement-based interaction design. *Human Technology* 12 (2016).

[34] Michael J Muller and Sara Kuhn. 1993. Special issue on participatory design. *Commun. ACM* 36, 6 (1993), 24–28.

[35] Bilge Mutlu and Jodi Forlizzi. 2008. Robots in organizations: the role of workflow, social, and environmental factors in human-robot interaction. In *Proceedings of the 3rd ACM/IEEE international conference on Human robot interaction*. ACM, 287–294.

[36] Hermann Daniel Neider. 2014. Applications of automata learning in verification and synthesis. (2014).

[37] Scott Niekum, Sachin Chitta, Andrew G Barto, Bhaskara Marthi, and Sarah Osentoski. 2013. Incremental Semantically Grounded Learning from Demonstration.. In *Robotics: Science and Systems*, Vol. 9. Berlin, Germany.

[38] Scott Niekum, Sarah Osentoski, George Konidaris, and Andrew G Barto. 2012. Learning and generalization of complex tasks from unstructured demonstrations. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 5239–5246.

[39] Jakob Nielsen. 1994. *Usability engineering*. Elsevier.

[40] Antti Oulasvirta, Esko Kurvinen, and Tomi Kankainen. 2003. Understanding contexts by being there: case studies in bodystorming. *Personal and ubiquitous computing* 7, 2 (2003), 125–134.

[41] Mike Phillips, Victor Hwang, Sachin Chitta, and Maxim Likhachev. 2016. Learning to plan for constrained manipulation from demonstrations. *Autonomous Robots* 40, 1 (2016), 109–124.

[42] David Porfirio, Allison Sauppé, Aws Albarghouthi, and Bilge Mutlu. 2018. Authoring and verifying human-robot interactions. In *The 31st Annual ACM Symposium on User Interface Software and Technology*. ACM, 75–86.

[43] Emmanuel Pot, Jérôme Monceaux, Rodolphe Gelin, and Bruno Maisonnier. 2009. Choregraphe: a graphical tool for humanoid robot programming. In *Robot and Human Interactive Communication, 2009. RO-MAN 2009. The 18th IEEE International Symposium on*. IEEE, 46–51.

[44] Rodrigo Pimentel. 2019. Chatito. (2019). `https://rodrigopivi.github.io/Chatito/`, Accessed: 2019-07-15.

[45] Stan Salvador and Philip Chan. 2007. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis* 11, 5 (2007), 561–580.

[46] Allison Sauppé and Bilge Mutlu. 2014. Design patterns for exploring and prototyping human-robot interactions. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*. ACM, 1439–1448.

[47] Dennis Schleicher, Peter Jones, and Oksana Kachur. 2010. Bodystorming as embodied designing. *Interactions* 17, 6 (2010), 47–51.

[48] Donald A Schön. 1987. Educating the reflective practitioner. (1987), 8–9.

[49] Donald A Schön. 1992. Designing as reflective conversation with the materials of a design situation. *Knowledge-based systems* 5, 1 (1992), 11.

[50] Phoebe Sengers, Kirsten Boehner, Shay David, and Joseph'Jofish' Kaye. 2005. Reflective design. In *Proceedings of the 4th decennial conference on Critical computing: between sense and sensibility*. ACM, 49–58.

[51] David Sirkin and Wendy Ju. 2014. Using embodied design improvisation as a design research tool. In *Proceedings of the international conference on Human Behavior in Design (HBiD 2014), Ascona, Switzerland*.

[52] David Sirkin and Wendy Ju. 2015. Embodied design improvisation: a method to make tacit design knowledge explicit and usable. In *Design Thinking Research*. Springer, 195–209.

[53] David Sirkin, Brian Mok, Stephen Yang, and Wendy Ju. 2015. Mechanical ottoman: how robotic furniture offers and withdraws support. In *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction*. ACM, 11–18.

[54] Alexander Skoglund, Boyko Iliev, Bourhane Kadmiry, and Rainer Palm. 2007. Programming by demonstration of pick-and-place tasks for industrial manipulators using task primitives. In *2007 International Symposium on Computational Intelligence in Robotics and Automation*. IEEE, 368–373.

[55] Brian K Smith. 2014. Bodystorming mobile learning experiences. *TechTrends* 58, 1 (2014), 71–76.

[56] Armando Solar-Lezama, Liviu Tancau, Rastislav Bodik, Sanjit Seshia, and Vijay Saraswat. 2006. Combinatorial sketching for finite programs. *ACM Sigplan Notices* 41, 11 (2006), 404–415.

[57] Marco Spadafora, Victor Chahuneau, Nikolas Martelaro, David Sirkin, and Wendy Ju. 2016. Designing the behavior of interactive objects. In *Proceedings of the TEI'16: Tenth International Conference on Tangible, Embedded, and Embodied Interaction*. ACM, 70–77.

[58] Steve Stemler. 2001. An overview of content analysis. *Practical assessment, research & evaluation* 7, 17 (2001), 137–146.

[59] Thalmic Labs. 2016. Myo Blog. (2016). `https://developerblog.myo.com/`, Accessed: 2019-07-15.

[60] Willemien Visser. 2010. Schön: Design as a reflective practice. *Collection* 2 (2010), 21–25.

[61] Ji-Dong Yim and Christopher D Shaw. 2009. Designing CALLY: a cell-phone robot. In *CHI'09 Extended Abstracts on Human Factors in Computing Systems*. ACM, 2659–2662.