Aalto University

OPEN ACCESS

This is an electronic reprint of the original article.
This reprint may differ from the original in pagination and typographic detail.

Racca, Mattia; Kyrki, Ville; Cakmak, Maya

Interactive Tuning of Robot Program Parameters via Expected Divergence Maximization

Published in:
Proceedings of the 2020 ACM/IEEE International Conference on Human-Robot Interaction, HRI'20

DOI:
10.1145/3319502.3374784

Published: 01/01/2020

Document Version
Peer-reviewed accepted author manuscript, also known as Final accepted manuscript or Post-print

Please cite the original version:
Racca, M., Kyrki, V., & Cakmak, M. (2020). Interactive Tuning of Robot Program Parameters via Expected Divergence Maximization. In *Proceedings of the 2020 ACM/IEEE International Conference on Human-Robot Interaction, HRI'20* (pp. 629–638). (ACM/IEEE International Conference on Human-Robot Interaction). IEEE. https://doi.org/10.1145/3319502.3374784

This material is protected by copyright and other intellectual property rights, and duplication or sale of all or part of any of the repository collections is not permitted, except that material may be duplicated by you for your research use or educational purposes in electronic or print form. You must obtain permission for any other use. Electronic or print copies may not be offered, whether for sale or otherwise to anyone who is not an authorised user.

Interactive Tuning of Robot Program Parameters via Expected Divergence Maximization

Mattia Racca
Aalto University
Espoo, Finland
mattia.racca@aalto.fi

Ville Kyrki
Aalto University
Espoo, Finland
ville.kyrki@aalto.fi

Maya Cakmak
University of Washington
Seattle, Washington, USA
mcakmak@cs.washington.edu

ABSTRACT

Enabling diverse users to program robots for different applications is critical for robots to be widely adopted. Most of the new collaborative robot manipulators come with intuitive programming interfaces that allow novice users to compose robot programs and tune their parameters. However, parameters like motion speeds or exerted forces cannot be easily demonstrated and often require manual tuning, resulting in a tedious trial-and-error process. To address this problem, we formulate tuning of one-dimensional parameters as an Active Learning problem where the learner iteratively refines its estimate of the feasible range of parameter values, by selecting informative queries. By executing the parametrized actions, the learner gathers the user's feedback, in the form of directional answers ("higher," "lower," or "fine"), and integrates it in the estimate. We propose an Active Learning approach based on Expected Divergence Maximization for this setting and compare it against two baselines with synthetic data. We further compare the approaches on a real-robot dataset obtained from programs written with a simple Domain-Specific Language for a robot arm and manually tuned by expert users (N=8) to perform four manipulation tasks. We evaluate the effectiveness and usability of our interactive tuning approach against manual tuning with a user study where novice users (N=8) tuned parameters of a human-robot hand-over program.

CCS CONCEPTS

• **Computing methodologies** → **Active learning settings**; • **Human-centered computing** → *User centered design*; • **Computer systems organization** → External interfaces for robotics.

KEYWORDS

Active Learning; End-User Programming; Human-Robot Interaction

ACM Reference Format:
Mattia Racca, Ville Kyrki, and Maya Cakmak. 2020. Interactive Tuning of Robot Program Parameters via Expected Divergence Maximization. In *Proceedings of the 2020 ACM/IEEE International Conference on Human-Robot Interaction (HRI '20), March 23–26, 2020, Cambridge, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3319502.3374784

1 INTRODUCTION

Programmability is the key advantage of robots over traditional manufacturing [44]. However, for robots to be widely adopted across industries and beyond structured manufacturing environments, it is critical for them to be programmable by a wide range of users with as little effort as possible. Growing research on End User Programming (EUP) for robotics aims to address this problem with novel user interfaces, programming languages, and techniques to aid or fully automate robot programming. Many of these approaches have been adopted by industrial robotics companies, such as Franka Emika or Rethink Robotics, whose robots come with intuitive programming interfaces.

Prior efforts have led to great solutions for end users to compose programs by combining discrete actions (e.g., [36, 47, 65, 66]), but tuning the parameters of those actions remains a tedious task. For example, consider a retail store robot that can be programmed to stock different items on shelves. Existing tools allow for the user to physically demonstrate this task or create programs from a sequence of actions like *move arm to target* or *close gripper* via visual programming interfaces. However, the parameters of these actions, such as how fast the arm should move or how much force the gripper should apply, cannot be easily demonstrated and need to be manually specified by the user. Specifying such parameters often requires the user to adopt a *trial-and-error* strategy, executing each action or the whole program a number of times while varying parameter values. This iterative and incremental process is needed for two reasons: first, to understand the impact of the parameter on the action execution, since a change of 0.01 meters per second or 1 Newton might not be instinctively meaningful, and second, to actually find an acceptable parameter value that satisfies the task's requirements. Moreover, finding a single parameter value that works is often not sufficient. Rather, users may want to identify a range of acceptable parameters to maintain flexibility in the program, in order to e.g., let the robot autonomously select an optimal value within the range based on the current external conditions.

Our work aims to address this challenge of tedious manual parameter tuning for robot programs. To this end, we propose an interactive tuning approach whereby the robot iteratively proposes parameter values and gathers the user's feedback to find a *range of feasible values* for one dimensional continuous parameters. We formulate this as an *Active Learning* (AL) [61] problem in which the learner iteratively refines its estimate of the feasible range as quickly and economically as possible by selecting informative query values, executing the parametrized actions with those values, and integrating the user feedback, in the form of *directional answers* ("higher", "lower", or "fine"), in its estimate. We adopt a Bayesian approach for this setting combined with Expected Divergence Maximization (an expected error reduction technique [58]) and compare it against two baseline algorithms both on synthetic and real-robot

HRI '20, March 23–26, 2020, Cambridge, United Kingdom
© 2020 Copyright held by the owner/author(s). Publication rights licensed to ACM. This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in *Proceedings of the 2020 ACM/IEEE International Conference on Human-Robot Interaction (HRI '20), March 23–26, 2020, Cambridge, United Kingdom*, https://doi.org/10.1145/3319502.3374784.

datasets. The real-robot dataset is obtained from expert programmers (N=8) using a simple *Domain-Specific Language* (DSL) for a robot arm (Franka Emika's Panda), manually tuning the program's parameters to perform four manipulation tasks. Finally, we evaluate the effectiveness and usability of our interactive tuning approach against manual tuning through a user study where novice users (N=8) tuned a hand-over program specified in our DSL.

2 RELATED WORK

Our work shares the goal of EUP for robots, which is to enable people with little or no programming background to be able to program robots with as little effort as possible. Work in this area has produced tools that allow novice users to combine robot primitives or actions to create complex programs, with input modalities ranging from visual programming [8, 31, 35, 36, 45, 46, 64, 65], kinesthetic teaching [1, 2, 60, 66], and natural language commands [13, 26, 33]. The robot actions available to compose programs are often parametrized, with number and complexity of the parameters depending on the level of abstraction adopted in the action design. While for certain parameters intuitive ways of specifying them have been proposed (e.g., specifying the goal pose of a robot motion with kinesthetic teaching), other parameters have to be either manually tuned via *Graphical User Interfaces* (GUIs) or simply set to default values. Our work addresses the problem of efficiently tuning parameters that are hard to demonstrate and therefore require the user to manually specify and refine them through trial-and-error. Along the same line, recent work in robot EUP aims to partially automate robot programming through program verification [37, 51, 56] and synthesis [39, 40, 50] to reduce the burden on the programmer.

One way that low-level motor skills can be programmed on a robot is through *Learning from Demonstration* (LfD) [5]. In contrast to high-level end-user programs, skills created through LfD are often not intended to be comprehended, modified or re-parametrized by end-users. Instead, LfD techniques require multiple demonstrations of the skill to learn a flexible model that works in different environments, either by modelling uncertainty over the demonstrated trajectories (e.g., [20, 21]) or by finding invariants in the demonstrations (e.g., [27, 48]). Similarly, our tuning framework addresses the problem of finding feasible ranges of parameters to allow for flexible programs, rather than simply finding a single parameter value that works. However, unlike LfD, our approach does not require the user to give multiple demonstrations but to provide sparse feedback on informative action executions.

The method developed in this work is an example of Active Learning (AL)–a machine learning paradigm in which the learner actively chooses its training data [4, 23, 61]. While passive supervised learning uses previously labeled data, active learners obtain labels from the user only for selected unlabeled samples. By selecting *informative* queries, active learners can reach desired performance levels more efficiently. This is particularly important for problems where collecting labels is costly, as in robotics and HRI where AL has seen growing interest. AL techniques have been used to economically learn robot policies [21, 22, 55, 62] and task representations [25, 34, 41, 53], to guide efficient information gathering [17], and to learn reward functions by querying the user with different type of queries [9, 10, 24, 59]. Another line of research has instead concentrated on the interactive nature of AL, with work investigating the design of active robot learners [18, 19], the ability of users to answer the robot's questions [32, 57] and the development of AL strategies that take into account the user in their query selection [11, 38, 54]. New types of queries, different from asking for a positive or negative label, have been adopted or developed through this line of research, including demonstration queries [16, 19], feature queries [10, 16, 19], and comparison queries [12, 59]. The approach we contribute makes use of a new type of query where the user, after evaluating a parametrized action execution, can either affirm the queried parameter value ("fine") or give feedback about the direction towards which the value should change ("higher" or "lower").

3 ACTIVE PARAMETER TUNING

In robot programming, parameter tuning is the process of searching for a value or a range of values of an action's parameter to make the action work as desired. For example, consider a robot arm picking an object from a retail store shelf. The amount of force exerted by the gripper, chosen to not harm packaging or crush its contents while still allowing its safe transport, depends on the specific item. It is unrealistic to expect programmers to tune such *force* value of the *grasping* action correctly without executing the action itself. Instead, programmers start out with a guess, test it, and possibly change value based on their observation. For example, if they observe that the robot's grasp crushes the packaging they can reduce the exerted force, whereas if it is too weak to lift the object they can increase it.

In this work, we assume that a parameter has a closed range of *feasible values* that allow the action to be successful or to meet the user's preferences in a specific context. Let $R = [L, U]$, with $L, U \in \mathbb{R}$, be the full range of possible values that a parameter v can assume. We define $F = [l, u] \subseteq R$ to be the range of *feasible values*. In some cases, finding a single feasible value that is within the feasible range (i.e. a value that makes the action succeed) might be sufficient. We are instead interested in identifying the full range of feasible parameter values i.e. estimating its lower and upper bounds, l and u. This allows programmers to create flexible programs that can autonomously adapt based on the specific situation. For example, a robot arm could select the speed of its motions inside the provided range based on the presence of people in its surroundings. The feasible range could also act as a pool of safe values for explorative techniques like Reinforcement Learning. Multiple versions of the same action can also be generated by sampling the feasible range, to increase the natural behaviour of social robots [30].

3.1 Bayesian estimation of parameter ranges

Given the incremental nature of parameter tuning process, we adopt a Bayesian approach. We model the prior probability of a parameter's value v to be correct independently of any particular task or user's preferences, as $p(v = x) \sim f_{\text{v}}(x)$, where $f_{\text{v}}(x)$ is a distribution suiting the nature of the parameter. This distribution can reflect available data on how the parameter is tuned in different contexts or incorporate prior knowledge such as safety indications (e.g., the robot should rarely go faster than a certain speed or exert more than a certain amount of force). In absence of any useful

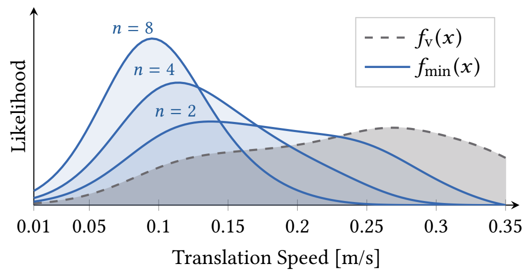


Figure 1: Comparison between $f_V(x)$, extracted from expert tuning for the *Translation Speed* parameter of the *Linear Motion* action, and three $f_{\min}(x)$, with different value of n.

information, an uninformative prior (where all possible outcomes have the same probability) can be used.

To estimate the bounds of the feasible range, we need $p(l = x)$ and $p(u = x)$, i.e. the probability of the lower bound and of the upper bound respectively to have value x. Using $f_V(x)$, we model these probabilities with the distribution of the sample minimum $f_{\min}(x)$ and of the sample maximum $f_{\max}(x)$ [49, 67] as

$$\begin{aligned} p(l = x) &\sim f_{\min}(x) = n(1 - F_V(x))^{n-1} f_V(x), \\ p(u = x) &\sim f_{\max}(x) = nF_V(x)^{n-1} f_V(x), \end{aligned} \tag{1}$$

where $F_V(x)$ is the cumulative distribution function of $f_V(x)$. Parameter n is the number of random variables sampled from $f_V(x)$ to estimate the sample minimum (or maximum). Figure 1 shows examples of $f_{\min}(x)$ with respect to a $f_V(x)$, with increasing values of n: as n increases, the probability mass moves towards L. For our estimation problem, n expresses how strongly we assume $f_V(x)$ to inform us about the distribution of l and u.

The computation of $f_{\min}(x)$ and $f_{\max}(x)$ may not be available in closed form for a generic $f_V(x)$. In this work, we approximate $f_V(x)$ and related distributions by discretization with their k-binned density histogram, making the problem tractable for any prior shape at the expense of extra computational load. Furthermore, we also discretize the parameter range R in k values, forming what will be the query pool Q for the AL method.

With $f_{\min}(x)$ and $f_{\max}(x)$ acting as prior distributions for l and u, we can adopt the active approach summarized in Algorithm 1. With query budget B being the available number of questions, the learning agent evaluates each value $v \in Q$ and selects the value to be queried q^*. Section 3.2.3 presents the selection process. Once q^* receives an answer a from the user, the learner computes the posteriors of both $f_{\min}(x)$ and $f_{\max}(x)$ (as presented in Section 3.2.2). Once the query budget is spent, the bounds $\hat{l}$ and $\hat{u}$ are computed as the mode of $f_{\min}(x)$ and $f_{\max}(x)$ respectively.

3.2 Active learning approach

In this section we present our design of queries and answers for the parameter range tuning problem and explain how information from a query and answer pair is integrated in the model. We then present our query selection strategy, along with two baseline strategies.

3.2.1 Queries and answers. We define a query as the execution of a particular action, parametrized with a parameter's value set to q^*.

Algorithm 1 Active Feasible Range Learning

```
Input: Query pool $Q$, $f_{\min}(x)$ and $f_{\max}(x)$, Query budget $B$
Output: Estimated feasible range $\hat{F} = [\hat{l}, \hat{u}]$
1:  while $B > 0$ do
2:      for all values $v \in Q$ do
3:          $S_v \leftarrow$ compute query score                [see Section 3.2.3]
4:      end for
5:      $q^* \leftarrow$ select query based on $S_v$              [see Equation 6]
6:      $a \leftarrow$ make selected query $q^*$ and wait for answer
7:      update $f_{\min}(x)$, $f_{\max}(x)$ given $q^*$ and $a$  [see Equation 3]
8:      $B \leftarrow B - 1$
9:  end while
10: $\hat{F} \leftarrow [\mathrm{argmax}_x f_{\min}(x),\ \mathrm{argmax}_x f_{\max}(x)]$
```

Based on the action execution, the user evaluates the queried value q^* and gives an answer a. Given that, for the addressed parameters, the range of possible values R is a subset of $\mathbb{R}$, our method expects three possible *directional answers*:

- The parameter value q^* is **fine**,
- The parameter value q^* must be **lower**,
- The parameter value q^* must be **higher**.

These directional answers require the user to evaluate the parameter, either by accepting it or by indicating if such value should be increased or decreased, avoiding the use of any numerical value by the user.

3.2.2 Model update. Once the answer a to query q^* is available, its information is integrated in the current model. We do so by computing the posterior distributions $f_{\min}(x|q^*, a)$ and $f_{\max}(x|q^*, a)$. Thanks to the discretized representation adopted, we can design *filter functions* as in [53] that modify the current priors by integrating the information coming from the question-answer pair. Suiting our directional queries, we use logistic functions $\lambda^{\pm}_{\phi,x_0}(x)$ as filters, defined as

$$\lambda^{\pm}_{\phi,x_0}(x) = \frac{1}{1 + e^{\pm\phi(x_0 - x)}}, \tag{2}$$

with x_0 being the midpoint of the function (where $\lambda^{\pm}_{\phi,x_0}(x) = 0.5$) and ϕ dictating the steepness of the curve at x_0. The $\lambda^{+}_{\phi,x_0}(x)$ goes from 0 to 1 as x grows; vice versa for $\lambda^{-}_{\phi,x_0}(x)$. Parameter ϕ can model the uncertainty of answers in the case of non-oracles, i.e. noisy users: for small values of ϕ, the filter function produces a less sharp update, retaining some probability mass for values close to q^* that were excluded by a possibly noisy answer. Unlike the method presented in [10], our approach does not however handle *I don't know* answers and requires the user to at least give a noisy answer to each query.

While there are three possible answers, a **fine** answer is equivalent to **lower** when updating $f_{\min}(x)$: if q^* is an acceptable value, then the lowerbound l must be lower. Similarly, a **fine** answer is equivalent to **higher** when updating $f_{\max}(x)$. Aside from this difference, query-answer pairs update $f_{\min}(x)$ and $f_{\max}(x)$ through

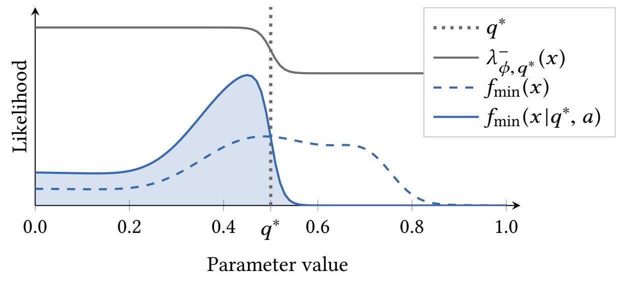


Figure 2: The posterior distribution $f_{\min}(x|q^*, a)$ is obtained with Equation 3 by updating the prior $f_{\min}(x)$ through $\lambda^-_{\phi,x_0}(x)$, after query q^* and answer $a =$ lower.

the filter function as

$$f_\cdot(x|q^*, a) = \begin{cases} \lambda^+_{\phi,q^*}(x) f_\cdot(x) & \text{if } a \text{ is } \textbf{higher} \text{ (or } \textbf{fine} \text{ if } f_{\max}(x)) \\ \lambda^-_{\phi,q^*}(x) f_\cdot(x) & \text{if } a \text{ is } \textbf{lower} \text{ (or } \textbf{fine} \text{ if } f_{\min}(x)) \end{cases} \tag{3}$$

with $f_\cdot(x)$ being either $f_{\min}(x)$ or $f_{\max}(x)$. After computation, the posterior is normalized and becomes the prior for successive queries. Figure 2 shows how posterior $f_{\min}(x|q^*, a)$ is computed by updating the $f_{\min}(x)$ prior after an answer $a =$ **lower**: the filter $\lambda^-_{\phi,x_0}(x)$ penalizes values greater then q^*, moving the probability mass according to the answer.

3.2.3 Query selection. To estimate the feasible parameter range $\hat{F}$, our learning approach iteratively selects parameter values from the query pool Q and integrates the related answers in the model. However, making queries in our scenario is costly: the robot needs to execute the action for each parameter value q^* to be queried, consuming valuable resources like the user's time and patience. Instance-based AL addresses this problem [29, 61]: the *best query* is chosen by estimating the queries' *informativeness* given the current model, consequently speeding up the learning.

For this work, we use an *Expected Divergence Maximization* strategy (**ExpDiv**): query informativeness is operationalized by measuring the *distance* between the prior model and the posterior resulting from a given query-answer pair. This AL strategy belongs to the broad family of expected error reduction techniques [58, 61].

In detail, for each parameter value $v \in Q$, we compute, for both distributions $f_{\min}(x)$ and $f_{\max}(x)$, the *expected divergence* between the prior distribution and the posterior distribution if v is queried: the more different the two distributions, the higher the divergence and, consequently, the information carried by the query. For each value $v \in Q$, two scores, $S_v^{\min}$ and $S_v^{\max}$, are computed as

$$\begin{aligned} S_v^{\cdot} &= \mathbb{E}_a[\mathbb{JS}(\overbrace{f_\cdot(x|v, a)}^{\text{post query}}, \overbrace{f_\cdot(x)}^{\text{pre query}})] \\ &= \sum_a p(a|v, f_\cdot(x)) \mathbb{JS}(f_\cdot(x|v, a), f_\cdot(x)) \end{aligned} \tag{4}$$

where $\mathbb{JS}$ denotes the Jensen-Shannon Divergence [43]. The divergence is *expected* because, at the time of the query selection, the answer a is not yet known: therefore the score is averaged over all possible answers. We estimate the probability of answer a given the queried value v and current model $p(a|v, f_\cdot(x))$ as

$$\begin{aligned} p(\textbf{lower}|v, f_\cdot(x)) &= \sum_{L \le x \le v} f_\cdot(x), \\ p(\textbf{higher}|v, f_\cdot(x)) &= 1 - p(\textbf{lower}|v, f_\cdot(x)). \end{aligned} \tag{5}$$

Computing $S_v^{\cdot}$ has complexity $O(k^2)$, where $k = |Q|$. With the scores computed, the learner selects the query q^* as

$$q^* = \underset{v}{\text{argmax}} \, \{S_v^{\min}, S_v^{\max}\}, \tag{6}$$

i.e. the value q^* that is expected to bring the most information to either posterior distribution. While other scores can be used (e.g., product of $S_v^{\min}$ and $S_v^{\max}$ or directly reasoning on the range distribution [49]), Equation 6 represent a simple yet effective option.

To evaluate the **ExpDiv** strategy, we adopt two strategies as baselines: a *random* strategy (**Random**) and a *split* strategy (**Split**). The former randomly selects its queries (complexity $O(1)$). The latter instead finds the value q^* that *splits* the prior distribution in two halves with similar probability mass (complexity $O(k)$). Strategy **Split** alternates between $f_{\min}(x)$ and $f_{\max}(x)$ and selects the value q^* as

$$q^* = \underset{v}{\text{argmin}} \left| \textstyle\sum_{x|L \le x \le v} f_\cdot(x) - \sum_{x|v \le x \le U} f_\cdot(x) \right|, \tag{7}$$

where $f_\cdot(x)$ is either $f_{\min}(x)$ or $f_{\max}(x)$. Strategy **Split** essentially acts as a weighted version of a binary search or, in AL terms, as an uncertainty sampling technique [42], since it queries the value v for which the answer is most uncertain, without considering the effects of the answer on the current model.

3.2.4 Cautious query selection. The three presented strategies do not directly take into account the likelihood of the queried value $f_v(q^*)$ in their selection process. When an informative prior is available (as in Experiment 2 of Section 4.2), a low likelihood $f_v(v)$ of parameter value v can be interpreted as v being *unsafe* (e.g., high grasping forces) or *useless* (e.g., very slow translation speeds). For obvious reasons, informative but unsafe queries should be avoided. Querying useless values should also be avoided, as previous research has shown how they can hinder the interaction and the usability of AL approaches [18, 54]. In line with recent work on risk-aware [15] and teacher-aware AL [11, 38, 54], we propose a *cautious* version of each presented strategy. Given a parameter τ, these cautious strategies constrain their choice to parameter values $v \in Q$ that belong to the τ% most likely values of $f_v(x)$, therefore avoiding unlikely values. For example, with $f_v(x)$ being a Normal distribution $\mathcal{N}(\mu, \sigma)$ and τ parameter being 95.4%, the cautious strategies would only query values that are within 2σ from μ.

4 EXPERIMENTS

We evaluate the proposed approach in three experiments. First, we compare the presented selection strategies based on a synthetic dataset of prior distributions $f_v(x)$. Second, we present a DSL for a compliant robot manipulator, consisting of 5 parametrized actions, and obtain priors from expert users (N=8). We evaluate our approach in this domain with an expert oracle. We finally investigate the usability of an active tuning interface against a passive interface in a study where novice users (N=8) tuned the feasible parameter ranges for a program specified in the proposed DSL.

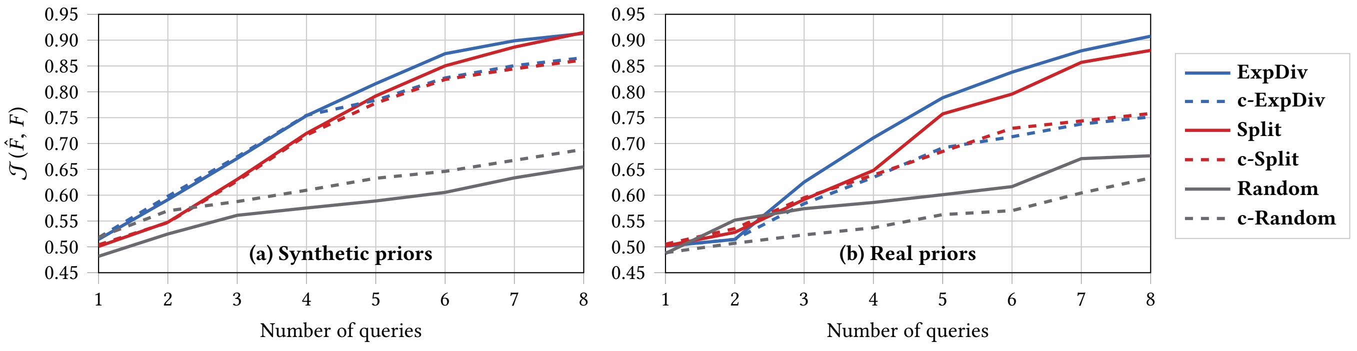


Figure 3: Mean Jaccard index $\mathcal{J}(\hat{F}, F)$ for proposed strategies and their cautious versions, averaged on (a) the synthetic dataset of 200 learning problems of Experiment 1 and on (b) the 44 learning problems of Experiment 2, with priors and ground truth ranges extracted from expert users' programs.

4.1 Experiment 1: Comparison with synthetic priors

To evaluate the proposed **ExpDiv** query selection strategy against the baselines and the cautious version of each strategy, we simulated the learning of feasible ranges F for a parameter $v \in [0, 1]$. We generated a dataset of 200 parameter value distributions $f_v(x)$, each with its own feasible range F acting as ground truth. Each distribution $f_v(x)$ consisted of a mixture of 4 Beta distributions $\mathcal{B}(\alpha, \beta)$. Each distribution's parameters α and β were sampled from a uniform distribution $\mathcal{U}(1, 20)$. Mixture weights were sampled from an uninformative Dirichlet distribution. Each $f_v(x)$ was then discretized, with $k = 100$ bins. The prior synthesis was designed to obtain multimodal priors and create a challenging benchmark. For each $f_v(x)$, we generated a true feasible range $F = [l, u]$ by sampling two values and assigning the lowest to l and the other to u. We refer to each pair of prior $f_v(x)$ and ground truth F as a *learning problem.*

We compared 6 learning strategies: **ExpDiv**, **Split** and **Random** presented in Section 3.2.3 and their cautious counterparts, **c-ExpDiv**, **c-Split** and **c-Random**. Each strategy had a budget B of 8 queries, with Equation 1's n set to 2, producing $f_{\min}(x)$ and $f_{\max}(x)$ distributions that do not heavily rely on $f_v(x)$. The answers were given by an oracle knowing the ground truth F. Parameters ϕ and τ were experimentally set to 200 and 85% respectively.

Results. To evaluate the strategies, we compute the *Jaccard index* after each question and related update as

$$\mathcal{J}(\hat{F}, F) = \frac{|\hat{F} \cap F|}{|\hat{F} \cup F|}, \tag{8}$$

measuring the overlap between the estimated range $\hat{F}$ and the ground truth F. Figure 3a shows the mean index for each strategy over the synthetic dataset. We run Friedman tests between the strategies' Jaccard indexes for each query number, paired by learning problem, finding significant differences already after 2 queries made (p<.01**). We therefore run pair-wise comparisons (Wilcoxon signed-rank test) on these indexes. While both strategies **ExpDiv** and **Split** perform better than **Random** (Wilcoxon signed-rank tests, all with p<.01**except after only one query), **ExpDiv** has a slightly higher mean $\mathcal{J}(\hat{F}, F)$ than **Split**, with statistically significant differences only after the second and third queries (Wilcoxon signed-rank, p<.05*). After more than four queries asked, strategy **Split**'s performance becomes comparable to **ExpDiv**'s one.

ExpDiv's higher performance comes however at a computational cost: **ExpDiv** and **c-ExpDiv** need on average 0.9 s to select a query, while the other strategies need less than a millisecond. While considerable in simulation, this difference in selection time is however negligible for our application, where lengthy robot actions are executed between query selections.

We can further observe how strategies **c-ExpDiv** and **c-Split** yield similar results to their incautious counterpart for the first 4 queries. After that, these strategies however converge to a lower mean index compared to strategies **ExpDiv** and **Split**. This is understandable, considering how their choice was constrained. To put their carefulness in context, 57% of **Random**'s queries, 17% of **ExpDiv**'s and 12% of **Split**'s belonged to the unlikely pool avoided by the cautious strategies.

Summarizing, results in simulation show how both strategies **ExpDiv** and **Split** can learn feasible parameter ranges in few queries and how their cautious versions differ from them for higher number of queries, sacrificing accuracy of the end results for safer queries.

4.2 Experiment 2: Comparison with real priors

In this section, we first present a simple DSL for programming a compliant robot arm. We then show how programs authored by expert users can be used to obtain priors $f_v(x)$ for each action's parameter and repeat the method comparison of Section 4.1 with such priors.

Proposed robot language. We designed a DSL for a compliant robot arm [52], with the 5 parametrized actions summarized in Table 1. Our robot arm of choice was a Franka Emika Panda (Figure 4a). Our DSL design closely follows the one of Desk, Franka Emika's proprietary EUP environment [28], abstracting the robot capabilities while ensuring a robust implementation and their intuitiveness for novice users. As in the Desk environment, programs are sequences of actions, with no branching nor looping. The programmer can add actions to the current program via a GUI. Action parameters can be instantiated by either kinesthetically moving the robot arm

Table 1: Proposed robot actions and relative parameters, with parameters used in Experiments 2 and 3 highlighted.

Action	Description	Parameters and their range of values
Linear Motion	The robot's end-effector (EE) moves* from the current pose to a **goal pose** with certain **translation speed**.	**goal pose**: SE(3) **translation speed**: [0.01, 0.35] m/s
Push Motion	The robot's EE moves* towards a **goal pose** with certain **translation speed** until it senses a contact force higher than a **force threshold**.	**goal pose**: SE(3) **translation speed**: [0.01, 0.25] m/s **force threshold**: [5.0, 20.0] N
Apply Force with Fingers	The robot's parallel gripper closes its fingers until a certain **grasping force** is exerted.	**grasping force**: [20.0, 100.0] N
Move Fingers	The robot's parallel gripper moves its fingers to a desired **finger distance**.	**finger distance**: [0.0, 0.08] m
User Synchronization	The robot's waits until the user exerts a force on it greater than a **force threshold** (e.g., by pushing or pulling it).	**force threshold**: [5.0, 30.0] N

* Linear interpolation with specified translation speed. The goal pose is defined in a frame at the robot's base.

(*goal pose* of motion actions) or via sliders on the GUI (all other parameters).

Expert programs. To evaluate the proposed framework in a real robotic scenario, we need value distributions $f_V(x)$ for the actions' parameters in our DSL. We designed four table-top tasks described in Table 2 and created programs expressed in the proposed DSL that perform them. We then randomized the parameter values of these programs and asked experts (graduate students in robotics, N=8) to tune them.

The experts tuned the programs with the GUI shown in Figure 4b. The GUI allowed the experts to select parameter values via sliders, execute robot actions, and revert them (i.e. bring the robot to the action's preconditions) in case further tuning was deemed necessary. To ensure variability in the collected data, the experts were instructed to adopt 2 different programming styles: a *safe style*, i.e. prioritizing safety of the robot and of possible humans close by, and an *efficient style*, i.e. minimizing task completion time.

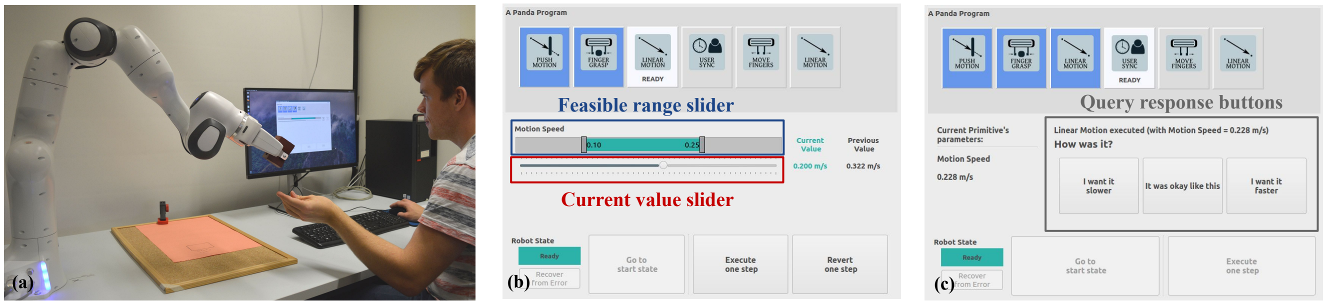
We collected a total of 64 expert authored programs (8 experts $\times$ 4 programs $\times$ 2 styles). For each action's parameter, we used *Kernel Density Estimation* (KDE) [63] on the values selected by the experts to learn two $f_V(x)$ priors, one for each programming style. As the goal of this experiment was to obtain the $f_V(x)$ priors, the experts did not specify a feasible range of values for the actions' parameters. We however computed expert ground truth ranges (for each parameter and action in each of the 4 programs) from the final single values selected by the experts in each style, considering the lower quartile and the upper quartile respectively as the lowerbound l and upperbound u. We then repeated the strategy comparison of Experiment 1 (same parameters for all strategies) on 44 learning problems made of expert ground truth ranges and the matching prior $f_V(x)$ (two learning problems for each parameter from the tasks of Table 2, one for each programming style).

Results. Figure 3b shows the mean $\mathcal{J}(\hat{F}, F)$ for each strategy. Significant differences between strategies are observed after 3 queries (Friedman test, p<.01**). We see how **ExpDiv** and **Split** outperforms **Random** already after 3 queries (Wilcoxon signed-rank, p<.01**). **ExpDiv** and **Split** have, as in Experiment 1, a similar performances, with a significant difference only after the third and forth query (p<.05*). The cautious strategies behave essentially as in Experiment 1, with strategies **c-ExpDiv** and **c-Split** having comparable results for low numbers of queries and later converging to a lower index, compared to their incautious counterparts.

Overall, the results show how directional queries and active selection strategies can obtain good range estimates with a low number of queries starting from priors learned from real programs, outperforming random selection. While the differences in performance between **ExpDiv** and **Split** are small in our scenario, we expect **ExpDiv** to better handle harder estimation problems (thanks to its *expected information gain* analysis). Like other expected error reduction techniques [61], **ExpDiv** is however more computationally expensive compared to simpler techniques like **Split**, making the choice between the two dependent on task-specific constraints.

Table 2: Benchmark tasks for Experiments 2 and 3.

Task	Program Description
Hand-over 6 actions, 7 parameters	The robot picks a soft toy, moves close to the user and waits for her to push/pull it. It then releases the toy and retreats.
Push against wall 3 actions, 4 parameters	The robot approaches a heavy box, pushes it against a wall, then retreats.
Push without tipping 3 actions, 4 parameters	The robot approaches a tall box, pushes it against a wall without tipping it, then retreats.
Transport without spilling 7 actions, 7 parameters	The robot moves close to user; waits for a spoon to be handed by pushing on the gripper. It then grasps the spoon which has a marble on it, transports the marble without dropping it on the table (2 motions), drops it into a bowl and retreats.



Linear Motion executed (with Motion Speed = 0.228 m/s)
How was it?

Figure 4: (a) Setup for Experiment 2 and 3 with a Panda robot mounted on a desk, with the user tuning robot programs through 2 GUIs: (b) a passive GUI that allows selection of parameter values to be tested on the robot and specification of the feasible range for each action's parameter, used in Experiment 2 and 3, and (c) a GUI that integrates the AL strategy c-ExpDiv, used in Experiment 3 for the active condition. Example of tuning procedure available at vimeo.com/mattiaracca/hri20.

4.3 Experiment 3: Evaluation with novices

To explore the usability of the proposed interactive tuning approach, we conducted a study with 8 novice users with no experience in robotics.

Experimental setup. The participants' task was to specify the feasible parameter ranges for a robot program in the proposed DSL for the hand-over task described in Table 2. Given the proximity of the participants to the robot arm and the interactive nature of the task, the robot's actions were implemented with impedance control. Furthermore, the range of values for each parameter reported in Table 1 was selected with the safety of the participants in mind. During the study, the participants directly controlled the robot through a GUI, with the experimenter only controlling the robot's emergency stop. This study was approved by Aalto University's Research Ethics Committee.

Conditions and protocol. Each participant specified the feasible parameter ranges for the hand-over program of Table 2 by using two different programming interfaces: a *passive interface*, shown in Figure 4b, and an *active interface*, embedding the proposed **c-ExpDiv** strategy (same parameters specified in Section 4.1 and priors learned on the safe version of the experts' programs) as shown in Figure 4c. The budget B was set to 3 queries as a tradeoff between performance and experiment duration. Each experiment lasted on average 40 minutes. The order of interfaces was counterbalanced.

Participants were instructed to find a range of feasible values according to their preferences for each parameter of the 6 actions composing the hand-over program. With the passive interface, the participants operated the robot as the experts did in Experiment 2, specifying the parameter values and the feasible ranges via sliders, with the possibility to freely execute and revert actions as many times as they wanted.

In the active condition, the GUI shown in Figure 4c guided the user through the tuning process, following a protocol based on Algorithm 1. The GUI goes through the program action by action and makes $B = 3$ queries about each parameter. First, **c-ExpDiv** selects a parameter value to be queried, announcing it to the user through the GUI (e.g., *"I will now execute this Linear Motion with speed 0.1 m/s"*). Second, the robot executes the action and the user's evaluation of the execution is requested via the GUI. Finally, either the robot's action is reverted to make further queries (if query budget is still available) or the tuning moves to the next action in the program. To aid the user's evaluation, the wording of the possible answers to the query was altered according to the addressed parameter, as shown in Figure 4c for the speed parameter.

Participants. Eight participants (age M=26, SD=3, 75% female) were recruited in a university campus. Participants had no hands-on experience with robotic arms or formal robotics education, with only two participants having an engineering background. Participants were rewarded with a movie ticket as compensation.

Logged data. We logged the wall-clock time the participants spent tuning the feasible ranges for each condition, along with the number of action executions needed with the passive interface. In the case of the active interface, the number of action executions was defined by the query budget B (21 total executions from 3 queries for each of the 7 parameters in the tuned program). After working on each interface, participants filled the SUS questionnaire [14]. At the end of the experiment, we gathered the participants' preferences regarding the interfaces along with free form feedback.

Results. We compare the participants' performance with active and passive parameter tuning in terms of efficiency and quality of tuning, as well as subjective usability. In particular, we measured the quality of the novices' parameter ranges by computing their Jaccard index against the expert ones from Experiment 2. The mean $\mathcal{J}(\hat{F}, F)$ achieved for the active condition after three queries was 0.52 (SD=0.28), against the 0.38 (SD=0.24) achieved in the passive condition after three executions (Wilcoxon signed-rank, T=463, p<.01**). While in the active condition the number of execution was capped to 21 by the query budget, with the passive condition the participants used an average of 27.5 (SD=9.5) total action executions. The mean $\mathcal{J}(\hat{F}, F)$ achieved with the passive interface at the end of the tuning remained however at 0.38.

Although these results do not translate to failures of the tuned program, we see how the novices specified parameter ranges substantially different from the expert ones of Experiment 2. The active interface however mitigated this tendency by constraining the user input through the querying mechanism.

Participants spent about 13 minutes tuning parameters in the passive condition and around 8 minutes in the active condition

(Wilcoxon signed-rank, T=2, p<.05*). Therefore, in the active condition participants obtained parameter ranges closer to the experts' ones in less time. Regarding usability, we observed a mean SUS score of 73.7 for the passive interface and of 73.1 for the active, suggesting *good usability* [7] for both interfaces.

In terms of subjective preference, three participants out of eight preferred the active interface. We summarize the participants' feedback about the interfaces in two categories: aid offered to the user and control over the tuning process. The active interface was generally described as easy to use, for its ability of guiding the tuning process and selecting the parameter values to try out (*"the active interface does the work for me"*). This aspect mattered especially at the beginning of the interaction, when the participants had less experience with the robot, the parameters, and the task at hand (*"the active interface is easier especially if you don't have any ideas about the values and parameters, but you have less control"* and *"I felt guided through the process"*). In comparison, the participants did not specifically comment on easiness of use or the helpfulness of the passive interface. Because of the safeness of the setup, the participants quickly realized they could try out parameter values with the passive interface without risk of damaging the robot or the environment. We believe this facilitated their exploration for good parameter values, making the guidance offered by the active interface less relevant.

The participants that preferred the passive interface backed their opinion mainly commenting on the freedom left to them compared to the active condition. Participants explained how the passive interface allowed them to achieve a more customized result (*"I had more freedom in choosing the parameter values"*) and to complete the task faster (*"I can choose parameters as I want, and I am much faster [than with the active interface]"*), even though they were actually slower. Participants might have felt that they were more efficient with the passive interface because they tended to reduce the number of executions per action over time, as they acquired experience with the different parameters. The number of executions went from a median of 10.5 executions for the first action to 3 executions for the last action of a program. Such adjustment over time was not present with the active interface, where the number of queries per parameter remained constant and there was no knowledge transfer from earlier actions to later ones.

Overall, although our active interface steered the novices' tuning and let them produce ranges closer to the expert ones with fewer action executions, the rigid control imposed represented its main weakness in this study. People's preference for being in control is in line with what observed for other embodied active learners [3, 18, 53]. This issue can be addressed by adopting a mixed-initiative strategy or having queries only *on-demand.* Alternatively, the proposed active approach could suggest parameter values to try out in more discrete ways like e.g., by highlighting parts of the GUI's sliders, while leaving full control to the user as in the passive interface.

5 LIMITATIONS AND FUTURE WORK

We have shown with the experiments our framework's potential for aiding novice programmers. Next, we discuss its limitations and possible ways of addressing them.

In our framework, both interfaces require the robot's ability to execute and revert actions to let the user to evaluate parameters. Although the active approach aims to reduce its action executions, operating a physical robot is still extremely time consuming. To speed up the tuning process, queries could be made via visualization and simulation tools, where robot and environment can be instantaneously reset, resorting to their execution on the physical robot only if needed (e.g., for actions that require user interaction, like the synchronization action of Table 1) or requested by the user.

While the passive interface lets the user decide which parameters to tune and whether to tune multiple of them at the same time, our active approach tunes parameters separately, one after the other, assuming their independence during the query selection process. While potentially reducing the number of queries, adapting our active approach to the simultaneous tuning of multiple parameters would require the modelling of the relationship between action's parameters in the priors, possibly requiring more data than in the single parameter case. Furthermore, the query selection process would become a multivariate optimization problem. The tuning of multiple parameters would also impact the interaction design. As research in LfD has shown how users provide better corrections when addressing one feature at a time [6], we need to investigate whether users can identify and evaluate the effects of the different parameters on the action execution, if several of them are tuned simultaneously. This could also impact the choice of query type: would users still be able to answer directional queries on single parameters or should simpler query types (e.g., label queries as *"was the overall action execution fine?"*) be adopted?

6 CONCLUSIONS

We tackled the problem of efficiently tuning parameters of robot actions, in the context of end-user robot programming. Our Active Learning approach aids the user in the complex and tedious task of specifying ranges of feasible values for action parameters by selecting values to be tested and estimating the ranges from the user's feedback. We evaluated the proposed approach, along with two baselines, both on synthetic and real-robot priors (obtained by implementing a simple domain-specific language for a robot arm and collecting expert programs targeting manipulations tasks). The results show how our active approach, together with the use of directional queries, allows for the estimation of parameter ranges in a small number of queries. Finally, we investigated the usability of our active approach with a user study, where novice users (N=8) tuned the parameter ranges of a real-robot hand-over program. Results show how novices were able to obtain feasible parameter ranges closer to the expert ones in less time with the active interface. However, participants' preference for the passive interface (5 out of 8), largely due to a perceived lack of control reported for the active interface, indicates the need for different ways of incorporating our strategy into end-user programming interfaces.

Acknowledgements. This work was supported by the Strategic Research Council at the Academy of Finland, decision 314180, and by the National Science Foundation (USA), awards IIS-1552427 "CAREER: End-User Programming of General-Purpose Robots" and IIS-1925043 "NRI: INT: COLLAB: Program Verification and Synthesis for Collaborative Robots."

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*.

[2] Sonya Alexandrova, Zachary Tatlock, and Maya Cakmak. 2015. RoboFlow: A flow-based visual programming language for mobile manipulation tasks. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 5537–5544.

[3] Saleema Amershi, Maya Cakmak, William B. Knox, and Todd Kulesza. 2014. Power to the people: The role of humans in interactive machine learning. *AI Magazine* 35, 4 (2014), 105–120.

[4] Dana Angluin. 1988. Queries and concept learning. *Machine Learning* 2, 4 (1988), 319–342.

[5] 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.

[6] Andrea Bajcsy, Dylan P. Losey, Marcia K. O'Malley, and Anca D. Dragan. 2018. Learning from physical human corrections, one feature at a time. In *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*. ACM, 141–149.

[7] Aaron Bangor, Philip T. Kortum, and James T. Miller. 2008. An empirical evaluation of the system usability scale. *International Journal of Human–Computer Interaction* 24, 6 (2008), 574–594.

[8] Emilia I. Barakova, Jan C. C. Gillesen, Bibi E. B. M. Huskens, and Tino Lourens. 2013. End-user programming architecture facilitates the uptake of robots in social therapies. *Robotics and Autonomous Systems* 61, 7 (2013), 704–713.

[9] Chandrayee Basu, Erdem Biyik, Zhixun He, Mukesh Singhal, and Dorsa Sadigh. 2019. Active Learning of Reward Dynamics from Hierarchical Queries. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

[10] Chandrayee Basu, Mukesh Singhal, and Anca D. Dragan. 2018. Learning from Richer Human Guidance: Augmenting Comparison-Based Learning with Feature Queries. In *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. ACM, 132–140.

[11] Aaron Bestick, Ravi Pandya, Ruzena Bajcsy, and Anca D. Dragan. 2018. Learning Human Ergornomic Preferences for Handovers. In *Robotics and Automation (ICRA), 2018 IEEE International Conference on*. IEEE, 3257–3264.

[12] Erdem Biyik and Dorsa Sadigh. 2018. Batch Active Preference-Based Learning of Reward Functions. In *Conference on Robot Learning*. 519–528.

[13] Michael Brenner, Nick Hawes, John D. Kelleher, and Jeremy L. Wyatt. 2007. Mediating between qualitative and quantitative representations for task-orientated human-robot interaction.. In *Proceedings of the International Joint Conference on Artificial Intelligence*. 2072–2077.

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

[15] Daniel S. Brown, Yuchen Cui, and Scott Niekum. 2018. Risk-Aware Active Inverse Reinforcement Learning. In *Conference on Robot Learning*. 362–372.

[16] Kalesha Bullard, Yannick Schroecker, and Sonia Chernova. 2019. Active learning within constrained environments through imitation of an expert questioner. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. AAAI Press, 2045–2052.

[17] Kalesha Bullard, Andrea L. Thomaz, and Sonia Chernova. 2018. Towards Intelligent Arbitration of Diverse Active Learning Queries. In *Intelligent Robots and Systems (IROS), 2018 IEEE/RSJ International Conference on*. IEEE, 6049–6056.

[18] Maya Cakmak, Crystal Chao, and Andrea L. Thomaz. 2010. Designing interactions for robot active learners. *IEEE Transactions on Autonomous Mental Development* 2, 2 (2010), 108–118.

[19] Maya Cakmak and Andrea L. Thomaz. 2012. Designing robot learners that ask good questions. In *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*. ACM, 17–24.

[20] Sylvain Calinon, Florent D'halluin, Eric L. Sauser, Darwin G Caldwell, and Aude G. Billard. 2010. Learning and reproduction of gestures by imitation. *IEEE Robotics & Automation Magazine* 17, 2 (2010), 44–54.

[21] Sonia Chernova and Manuela Veloso. 2007. Confidence-based policy learning from demonstration using gaussian mixture models. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and MultiAgent Systems*. ACM, 233.

[22] Sonia Chernova and Manuela Veloso. 2009. Interactive policy learning through confidence-based autonomy. *Journal of Artificial Intelligence Research* 34, 1 (2009), 1.

[23] David A. Cohn, Zoubin Ghahramani, and Michael I. Jordan. 1996. Active learning with statistical models. *Journal of Artificial Intelligence Research* 4, 1 (1996), 129–145.

[24] Yuchen Cui and Scott Niekum. 2018. Active reward learning from critiques. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 6907–6914.

[25] Joachim de Greeff and Tony Belpaeme. 2015. Why robots should be social: Enhancing machine learning through social human-robot interaction. *PLoS one* 10, 9 (2015), e0138061.

[26] Felix Duvallet, Thomas Kollar, and Anthony Stentz. 2013. Imitation learning for natural language direction following through unknown environments. In *2013 IEEE International Conference on Robotics and Automation*. IEEE, 1047–1053.

[27] Staffan Ekvall and Danica Kragic. 2008. Robot learning from demonstration: a task-level planning approach. *International Journal of Advanced Robotic Systems* 5, 3 (2008), 33.

[28] FRANKA EMIKA. 2019. *Panda's Capability*. https://www.franka.de/capability/

[29] Yifan Fu, Xingquan Zhu, and Bin Li. 2013. A survey on instance selection for active learning. *Knowledge and information systems* 35, 2 (2013), 249–283.

[30] Michael J. Gielniak, C. Karen Liu, and Andrea L. Thomaz. 2011. Task-aware variations in robot motion. In *2011 IEEE International Conference on Robotics and Automation*. IEEE, 3921–3927.

[31] Dylan F. Glas, Takayuki Kanda, and Hiroshi Ishiguro. 2016. Human-robot interaction design using Interaction Composer eight years of lessons learned. In *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 303–310.

[32] Victor Gonzalez-Pacheco, Maria Malfaz, Alvaro Castro-Gonzalez, Jose C. Castillo, Fernando Alonso-Martín, and Miguel A. Salichs. 2018. Analyzing the Impact of Different Feature Queries in Active Learning for Social Robots. *International Journal of Social Robotics* 10, 2 (2018), 251–264.

[33] Javi F. Gorostiza and Miguel A. Salichs. 2011. End-user programming of a social robot by dialog. *Robotics and Autonomous Systems* 59, 12 (2011), 1102–1114.

[34] Bradley Hayes and Brian Scassellati. 2014. Discovering task constraints through observation and active learning. In *Intelligent Robots and Systems (IROS), 2014 IEEE/RSJ International Conference on*. IEEE, 4442–4449.

[35] 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.

[36] Justin Huang, Tessa Lau, and Maya Cakmak. 2016. Design and evaluation of a rapid programming system for service robots. In *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 295–302.

[37] Benjamin Johnson and Hadas Kress-Gazit. 2012. Probabilistic Guarantees for High-Level Robot Behavior in the Presence of Sensor Error. *Autonomous Robots* 33 (2012), 309–321.

[38] Taylor Kessler Faulkner, Reymundo A. Gutierrez, Elaine Schaertl Short, Guy Hoffman, and Andrea L. Thomaz. 2019. Active Attention-Modified Policy Shaping. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*. 728–736.

[39] Hadas Kress-Gazit, Morteza Lahijanian, and Vasumathi Raman. 2018. Synthesis for robots: Guarantees and feedback for robot behavior. *Annual Review of Control, Robotics, and Autonomous Systems* 1 (2018), 211–236.

[40] Alap Kshirsagar, Hadas Kress-Gazit, and Guy Hoffman. 2019. Specifying and Synthesizing Human-Robot Handovers. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*.

[41] Johannes Kulick, Marc Toussaint, Tobias Lang, and Manuel Lopes. 2013. Active Learning for Teaching a Robot Grounded Relational Symbols. In *Proceedings of the International Joint Conference on Artificial Intelligence*. 1451–1457.

[42] David D. Lewis and William A. Gale. 1994. A sequential algorithm for training text classifiers. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. Springer-Verlag, 3–12.

[43] Jianhua Lin. 1991. Divergence measures based on the Shannon entropy. *IEEE Transactions on Information Theory* 37, 1 (1991), 145–151.

[44] Tomas Lozano-Perez. 1983. Robot Programming. *Proc. IEEE* 71, 7 (1983), 821–841.

[45] Marco Manca, Fabio Paternò, and Carmen Santoro. 2019. Analyzing Trigger-Action Programming for Personalization of Robot Behaviour in IoT Environments. In *International Symposium on End User Development*. Springer, 100–114.

[46] Carlos Mateo, Alberto Brunete, Ernesto Gambao, and Miguel Hernando. 2014. Hammer: An Android based application for end-user industrial robot programming. In *2014 IEEE/ASME 10th International Conference on Mechatronic and Embedded Systems and Applications (MESA)*. IEEE, 1–6.

[47] Hai Nguyen, Matei Ciocarlie, Kaijen Hsiao, and Charles C. Kemp. 2013. Ros commander (ROSCo): Behavior creation for home robots. In *2013 IEEE International Conference on Robotics and Automation*. IEEE, 467–474.

[48] 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.

[49] Steve Paik. 2015. *On the distribution of the range of a sample*. Technical Report.

[50] David Porfirio, Evan Fisher, Allison Sauppé, Aws Albarghouthi, and Bilge Mutlu. 2019. Bodystorming Human-Robot Interactions. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*.

[51] 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.

[52] Mattia Racca. 2019. EUPanda: a End-User Programming Framework for the FRANKA EMIKA Panda. https://github.com/MattiaRacca/eupanda.

[53] Mattia Racca and Ville Kyrki. 2018. Active Robot Learning for Temporal Task Models. In *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. ACM, 123–131.

[54] Mattia Racca, Antti Oulasvirta, and Ville Kyrki. 2019. Teacher-Aware Active Robot Learning. In *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 335–343.

[55] Nemanja Rakicevic and Petar Kormushev. 2019. Active learning via informed search in movement parameter space for efficient robot task learning and transfer. *Autonomous Robots* (2019), 1–19.

[56] Vasumathi Raman and Hadas Kress-Gazit. 2011. Analyzing unsynthesizable specifications for high-level robot behavior using LTLMoP. In *International Conference on Computer Aided Verification*. Springer, 663–668.

[57] Stephanie Rosenthal, Manuela Veloso, and Anind K. Dey. 2012. Acquiring accurate human responses to robots' questions. *International Journal of Social Robotics* 4, 2 (2012), 117–129.

[58] Nicholas Roy and Andrew Mccallum. 2001. Toward Optimal Active Learning through Sampling Estimation of Error Reduction. In *Proceedings of 18th International Conference on Machine Learning*.

[59] Dorsa Sadigh, Anca Dragan, Shankar S. Sastry, and Sanjit A. Seshia. 2017. Active Preference-Based Learning of Reward Functions. In *Robotics: Science and Systems*.

[60] Casper Schou, Jens S. Damgaard, Simon Bøgh, and Ole Madsen. 2013. Human-robot interface for instructing industrial tasks using kinesthetic teaching. In *IEEE ISR 2013*. IEEE, 1–6.

[61] Burr Settles. 2012. Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 6, 1 (2012), 1–114.

[62] Elaine Schaertl Short, Adam Allevato, and Andrea L. Thomaz. 2019. SAIL: Simulation-Informed Active In-the-Wild Learning. In *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 468–477.

[63] Bernard W. Silverman. 1986. Density Estimation for Statistics and Data Analysis. *Monographs on Statistics and Applied Probability* 26 (1986).

[64] Franz Steinmetz and Roman Weitschat. 2016. Skill parametrization approaches and skill architecture for human-robot interaction. In *2016 IEEE International Conference on Automation Science and Engineering (CASE)*. IEEE, 280–285.

[65] Franz Steinmetz, Annika Wollschläger, and Roman Weitschat. 2018. RAZER-A Human-Robot Interface for Visual Task-Level Programming and Intuitive Skill Parameterization. *IEEE Robotics and Automation Letters* 3, 3 (2018), 1362–1369.

[66] Maj Stenmark, Mathias Haage, and Elin Anna Topp. 2017. Simplified programming of re-usable skills on a safe industrial robot: Prototype and evaluation. In *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*. ACM, 463–472.

[67] Samuel S. Wilks. 1943. *Mathematical Statistics*. Princeton University Press.