Beyond Geometric Path Planning: Learning Context-Driven Trajectory Preferences via Sub-optimal Feedback

Ashesh Jain, Shikhar Sharma and Ashutosh Saxena

Abstract We consider the problem of learning preferences over trajectories for mobile manipulators such as personal robots and assembly line robots. The preferences we learn are more intricate than those arising from simple geometric constraints on robot's trajectory, such as distance of the robot from human etc. Our preferences are rather governed by the surrounding context of various objects and human interactions in the environment. Such preferences makes the problem challenging because the criterion of defining a good trajectory now varies with the task, with the environment and across the users. Furthermore, demonstrating optimal trajectories (e.g., learning from expert's demonstrations) is often challenging and non-intuitive on high degrees of freedom manipulators. In this work, we propose an approach that requires a *non-expert* user to only incrementally improve the trajectory currently proposed by the robot. We implement our algorithm on two high degree-of-freedom robots, PR2 and Baxter, and present three intuitive mechanisms for providing such incremental feedback. In our experimental evaluation we consider two context rich settings – household chores and grocery store checkout – and show that users are able to train the robot with just a few feedbacks (taking only a few minutes). Despite receiving sub-optimal feedback from non-expert users, our algorithm enjoys theoretical bounds on regret that match the asymptotic rates of optimal trajectory algorithms.

1 Introduction

Recent advances in robotics have resulted in mobile manipulators with high degree of freedom (DoF) arms. However, the use of high DoF arms has so far been largely successful only in structured environments such as manufacturing scenarios where they perform same repetitive motions (e.g., recent deployment of Baxter on assembly lines). A major challenge in the deployment of these robots in unstructured

Ashesh Jain[*], Shikhar Sharma[†], Ashutosh Saxena[*]
[*]Department of Computer Science, Cornell University, Ithaca, New York.
[†]Indian Institute of Technology Kanpur, India.
e-mail: `{ashesh,ss2986,asaxena}@cs.cornell.edu`
[†]This work was done when Sharma was an intern at the Cornell University.

environments (such as a grocery checkout counter or in our homes) is their lack of understanding of user preferences and thereby not producing desirable motions. In this work we address the problem of learning preferences over trajectories for high DoF robots such as Baxter or PR2. We consider a variety of household chores for PR2 and grocery store checkout tasks for Baxter.

A key problem for high DoF manipulators lies in identifying an appropriate trajectory for a task. An appropriate trajectory not only needs to be valid from a geometric point (i.e., feasible and obstacle-free, the criterion that most path planners focus on), but it also needs to satisfy the user's preferences. Such users' preferences over trajectories vary between users, between tasks, and between the environments the trajectory is performed in. For example, a household robot should move a glass of water in an upright position without jerks while maintaining a safe distance from nearby electronic devices. In another example, a robot checking out a santoku knife[1] at a grocery store should strictly move it at a safe distance from nearby humans. Furthermore, straight-line trajectories in Euclidean space may no longer be the preferred ones. For example, trajectories of heavy items should not pass over fragile items but rather move around them. These preferences are often hard to describe and anticipate without knowing where and how the robot is deployed. This makes it infeasible to manually encode (e.g., [26]) them in existing path planners (e.g., [10, 39, 44]) a priori.

We learn user preferences over trajectories via eliciting sub-optimal suggestions from the user for improving a trajectory. Unlike in other learning settings, where an expert first demonstrates optimal trajectories [4] for a task to the robot, our learning model does not rely on the user's ability to demonstrate optimal trajectories a priori. Instead, our learning algorithm explicitly guides the learning process and merely requires the user to incrementally improve the robots trajectories thereby learning user preferences and not that of expert's. This procedure of learning from sub-optimal suggestions is known as coactive learning and has been previously studied in information retrieval [41]. We contribute by introducing this new method of learning to the robotics community and highlight its advantages over learning from demonstration for high DoF robots. We build a system to realize this learning algorithm on PR2 and Baxter robots, and also leverage the robot specific design to allow users easily give preference feedback required by our algorithm.

Our experiments show that a robot trained using this approach can autonomously perform new tasks and if need be, only a small number of interactions are sufficient to tune the robot to the new task. *Since the user does not have to demonstrate a (near) optimal trajectory to the robot*, the feedback is easier to provide and more widely applicable. Nevertheless, it leads to an online learning algorithm with provable regret bounds that decay at the same rate as for optimal demonstrations.

In our empirical evaluation, we learn preferences for a two high DoF robots, PR2 and Baxter, on a variety of household and grocery checkout tasks respectively. Using the expressive trajectory features from our previous work [19], we show how our algorithm learns preferences from online user feedback on a broad range of tasks for which object properties are of particular importance (e.g., manipulating sharp objects with humans in the vicinity). We extensively evaluate our approach on

[1] A kitchen knife originating in Japan.

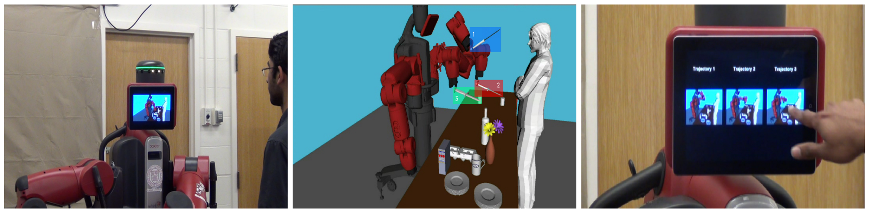


Fig. 1: **Re-rank feedback mechanism: (Left)** Robot ranks trajectories using the score function and **(Middle)** displays top three trajectories on a touch screen device (iPad here). **(Right)** As feedback, the user improves the ranking by selecting the third trajectory.

a set of 35 household tasks and 16 grocery checkout tasks, both in batch experiments as well as through robotic experiments wherein users provide their preferences on the robot. Our results show that our system not only quickly learns good trajectories on individual tasks, but also generalizes well to tasks that the algorithm has not seen before. We now describe the learning setting along with mechanisms for eliciting preference feedback and highlight attributes of Baxter robot that makes it well suited for our algorithm.

2 Coactive learning with incremental feedback

We propose an online algorithm for learning preferences in trajectories from sub-optimal user feedback. At each step robot receives a task as input and outputs a trajectory that maximizes its current estimate of some score function. It then observes a user feedback – an improved trajectory – and updates the score function to better match the user preferences. This procedure of learning via iterative improvement is known as coactive learning.

Our goal is to even learn from the feedback given by non-expert users. We therefore require the feedback to only be *incrementally* better (as compared to being close to optimal) in expectation, and will show that such feedback is sufficient for the algorithm's convergence. It is in contrast to learning from demonstration (LfD) methods [1, 25, 36, 37] which require (near) optimal kinesthetic demonstrations of the complete trajectory. Such demonstrations can be extremely challenging and non-intuitive to provide for many high DoF manipulators [2]. Instead, we found that it is more intuitive for users to give an incremental feedback on certain high DoF arms such as Barrett WAM and Baxter. With a zero-force gravity-compensation (zero-G) mode, the robot arms becomes light and the users can effortlessly steer them to desired configuration. On Baxter, this zero-G mode is automatically activated when a user holds the robot's wrist (see Figure 2, middle). We use this zero-G mode as a feedback method for incrementally improving the trajectory by correcting a

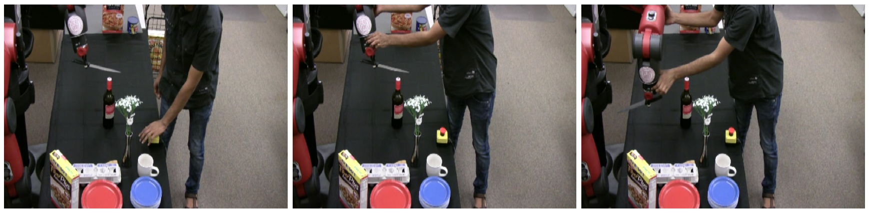


Fig. 2: **Zero-G feedback mechanism:** Robot checking out items in a grocery store moves knife close to human. **(Left)** User stops the trajectory and **(Middle)** activates the zero-G mode by holding Baxter's wrist. **(Right)** User then improves the trajectory by moving the knife away and rotating it.

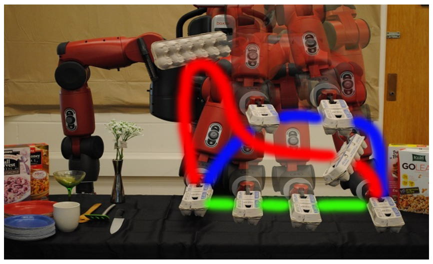


Fig. 3: **Re-ranking feedback:** Shows three trajectories for moving egg carton from left to right. Using the current estimate of score function robot ranks them as red, green and blue. As feedback user clicks the green trajectory. **Preference**: Eggs are fragile. They should be kept upright and near the supporting surface.

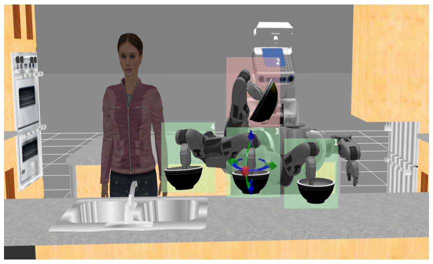


Fig. 4: **Interactive feedback.** Task here is to move a bowl filled with water. The robot presents a bad trajectory with waypoints 1-2-4 to the user. As feedback user moves waypoint 2 (red) to waypoint 3 (green) using Rviz interactive markers. The interactive markers guides the user to correct the waypoint.

waypoint. We now summarize three feedback mechanisms that enable the user to iteratively provide improved trajectories.
(a) Re-ranking: We display the ranking of trajectories using OpenRAVE [12] on a touch screen device and ask the user to identify whether any of the lower-ranked trajectories is better than the top-ranked one. User sequentially observes the trajectories in order of their current predicted scores and clicks on the first trajectory which is better than the top ranked trajectory. Figure 1 shows three trajectories for moving knife. As feedback user moves the trajectory at rank 3 to the top position. Likewise, Figure 3 shows three trajectories for moving an egg carton. Using the current estimate of score function robot ranks them as red (1^{st}), green (2^{nd}) and blue (3^{rd}). Since eggs are fragile user moves green trajectory to the top position.

(b) Zero-G: This feedback allows the user to correct trajectory waypoints by physically changing robot's arm configuration as shown in Figure 2. This feedback is useful (i) for bootstrapping the robot, (ii) for avoiding local maxima where the top trajectories in the ranked list are all bad but ordered correctly, and (iii) when the user is satisfied with the top ranked trajectory except for minor errors. A counterpart of this feedback is keyframe based LfD [2] where an expert demonstrates a sequence of optimal waypoints instead of the complete trajectory.

(c) Interactive: For the robots whose hardware does not permit zero-G feedback, such as PR2, we built an alternative interactive Rviz-ROS [16] interface for allowing the users to improve the trajectories by waypoint correction. Figure 4 shows a robot moving a bowl with one bad waypoint (in red), and the user provides a feedback by correcting it. This feedback serves the same purpose as zero-G but it's elicited via simulator.
Note that in all three kinds of feedback, the user never reveals the optimal trajectory to the algorithm but just provides a slightly improved trajectory (in expectation).

3 Learning and Feedback Model

We model the learning problem in the following way. For a given task, the robot is given a context x that describes the environment, the objects, and any other input relevant to the problem. The robot has to figure out what is a good trajectory y for this context. Formally, we assume that the user has a scoring function $s^*(x,y)$ that reflects how much he values each trajectory y for context x. The higher the score, the better the trajectory. Note that this scoring function cannot be observed directly, nor do we assume that the user can actually provide cardinal valuations according to this function. Instead, we merely assume that the user can provide us with *preferences* that reflect this scoring function. The robot's goal is to learn a function $s(x,y;w)$ (where w are the parameters to be learned) that approximates the user's true scoring function $s^*(x,y)$ as closely as possible.

Interaction Model. The learning process proceeds through the following repeated cycle of interactions.

Step 1: The robot receives a context x and uses a planner to sample a set of trajectories, and ranks them according to its current approximate scoring function $s(x,y;w)$.

Step 2: The user either lets the robot execute the top-ranked trajectory, or corrects the robot by providing an improved trajectory $\bar{y}$. This provides feedback indicating that $s^*(x,\bar{y}) > s^*(x,y)$.

Step 3: The robot now updates the parameter w of $s(x,y;w)$ based on this preference feedback and returns to step 1.

Regret. The robot's performance will be measured in terms of regret, $REG_T = \frac{1}{T}\sum_{t=1}^{T}[s^*(x_t,y_t^*) - s^*(x_t,y_t)]$, which compares the robot's trajectory y_t at each time step t against the optimal trajectory y_t^* maximizing the user's unknown scoring function $s^*(x,y)$, $y_t^* = argmax_y s^*(x_t,y)$. Note that the regret is expressed in terms of the user's true scoring function s^*, even though this function is *never observed*. Regret characterizes the performance of the robot over its whole lifetime, therefore reflecting how well it performs *throughout* the learning process. We will employ learning algorithms with theoretical bounds on the regret for scoring functions that are linear in their parameters, making only minimal assumptions about the difference in score between $s^*(x,\bar{y})$ and $s^*(x,y)$ in Step 2 of the learning process.

4 Learning Algorithm

For each task, we model the user's scoring function $s^*(x,y)$ with the following parametrized family of functions.

$$s(x,y;w) = w \cdot \phi(x,y) \tag{1}$$

w is a weight vector that needs to be learned, and $\phi(\cdot)$ are features describing trajectory y for context x. We further decompose the score function in two parts, one only concerned with the objects the trajectory is interacting with, and the other with the object being manipulated and the environment

$$s(x,y;w_O,w_E) = s_O(x,y;w_O) + s_E(x,y;w_E) = w_O \cdot \phi_O(x,y) + w_E \cdot \phi_E(x,y) \tag{2}$$

For more details on the features $\phi_O(\cdot)$ and $\phi_E(\cdot)$, we refer the readers to our previous work Jain et. al. [19].

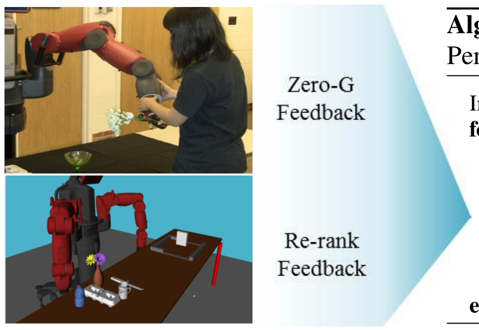
4.1 Computing Trajectory Rankings

For obtaining the top trajectory (or a top few) for a given task with context x, we would like to maximize the current scoring function $s(x,y;w_O,w_E)$.

$$y^* = \arg\max_y s(x,y;w_O,w_E). \tag{3}$$

Second, for a given set $\{y^{(1)},\ldots,y^{(n)}\}$ of discrete trajectories, we need to compute (3). Fortunately, the latter problem is easy to solve and simply amounts to sorting the trajectories by their trajectory scores $s(x,y^{(i)};w_O,w_E)$. Two effective ways of solving the former problem is either discretizing the state space or directly sampling trajectories from the continuous space. Previously both approaches [3, 6, 7, 11, 46] have been studied. However, for high DoF manipulators sampling based approaches [6, 11] maintain tractability of the problem, hence we take this approach. More precisely, similar to Berg et al. [6], we sample trajectories using rapidly-exploring random tree (RRT) [27].[2] Since our primary goal is to learn a score function on trajectories we now describe our learning algorithm and for more details on sampling trajectories we refer interested readers to [15, 17].

4.2 Learning the Scoring Function



Algorithm 1 Trajectory Preference Perceptron. (TPP)

```
Initialize $w_O^{(1)} \leftarrow 0$, $w_E^{(1)} \leftarrow 0$
for $t = 1$ to $T$ do
    Sample trajectories $\{y^{(1)},...,y^{(n)}\}$
    $y_t = argmax_y s(x_t,y;w_O^{(t)},w_E^{(t)})$
    Obtain user feedback $\bar{y}_t$
    $w_O^{(t+1)} \leftarrow w_O^{(t)} + \phi_O(x_t,\bar{y}_t) - \phi_O(x_t,y_t)$
    $w_E^{(t+1)} \leftarrow w_E^{(t)} + \phi_E(x_t,\bar{y}_t) - \phi_E(x_t,y_t)$
end for
```

Fig. 5: Shows our system design, for grocery store settings, which provides users with three choices for iteratively improving trajectories. In one type of feedback (**zero-G** or **interative** feedback in case of PR2) user corrects a trajectory waypoint directly on the robot while in the second (**re-rank**) user chooses the top trajectory out of 5 shown on the simulator.

The goal is to learn the parameters w_O and w_E of the scoring function $s(x,y;w_O,w_E)$ so that it can be used to rank trajectories according to the user's preferences. To do so, we adapt the Preference Perceptron algorithm [41] as detailed in Algorithm 1, and we call it the Trajectory Preference Perceptron (TPP). Given a context x_t, the top-ranked trajectory y_t under the current parameters w_O and w_E, and the user's feedback trajectory $\bar{y}_t$, the TPP updates the weights in the direction $\phi_O(x_t,\bar{y}_t) - \phi_O(x_t,y_t)$ and $\phi_E(x_t,\bar{y}_t) - \phi_E(x_t,y_t)$ respectively. Figure 5 shows an overview of our system design.

Despite its simplicity and even though the algorithm typically does not receive the optimal trajectory $y_t^* = \arg\max_y s^*(x_t,y)$ as feedback, the *TPP enjoys guarantees on the regret* [41]. We merely need to characterize by how much the feedback improves on the presented ranking using the following definition of expected α-informative feedback: $E_t[s^*(x_t,\bar{y}_t)] \geq s^*(x_t,y_t) + \alpha(s^*(x_t,y_t^*) - s^*(x_t,y_t)) - \xi_t$. This definition states that the user feedback should have a score of $\bar{y}_t$ that is – in expectation over the users choices – higher than that of y_t by a fraction $\alpha \in (0,1]$

[2] When RRT becomes too slow, we switch to a more efficient bidirectional-RRT.The cost function (or its approximation) we learn can be fed to trajectory optimizers like CHOMP [39] or optimal planners like RRT* [23] to produce reasonably good trajectories.

of the maximum possible range $s^*(x_t,\bar{y}_t) - s^*(x_t,y_t)$. If this condition is not fulfilled due to bias in the feedback, the slack variable ξ_t captures the amount of violation. In this way any feedback can be described by an appropriate combination of α and ξ_t. Using these two parameters, the proof by Shivaswamy and Joachims [41] can be adapted to show that average regret of TPP is upper bounded by $E[REG_T] \leq \mathcal{O}(\frac{1}{\alpha\sqrt{T}} + \frac{1}{\alpha T}\sum_{t=1}^{T}\xi_t)$.

5 Related Work

Teaching a robot to produce desired motions has been a long standing goal and several approaches have been studied. Most of the past research has focussed on mimicking expert's demonstrations, for example, autonomous helicopter flights [1], ball-in-a-cup experiment [25], planning 2-D paths [36, 37], etc. Such settings (learning from demonstration, LfD) assume that kinesthetic demonstrations are intuitive to an end-user and it is clear to an expert what constitutes a good trajectory. In many scenarios, especially involving high DoF manipulators, this is extremely challenging to do [2].[3] This is because the users have to give not only the end-effector's location at each time-step, but also the full configuration of the arm in a spatially and temporally consistent manner. In our setting, the user never discloses the optimal trajectory (or provide optimal feedback), but instead, the robot learns preferences from sub-optimal suggestions for how the trajectory can be improved.

Some later works in LfD provided ways for handling noisy demonstrations, under the assumption that demonstrations are either near optimal [48] or locally optimal [29]. Providing noisy demonstrations is different from providing relative preferences, which are biased and can be far from optimal. We compare with an algorithm for noisy LfD learning in our experiments. A recent work [47] leverages user feedback to learn rewards of a Markov decision process. Our approach advances over [47] and Calinon et. al. [9] in that it models sub-optimality in user feedback and theoretically converges to user's hidden score function. We also capture the necessary contextual information for household and grocery store robots, while such context is absent in [9, 47]. Our application scenario of learning trajectories for high DoF manipulations performing tasks in presence of different objects and environmental constraints goes beyond the application scenarios that previous works have considered. We use appropriate features that consider robot configurations, object-object relations, and temporal behavior, and use them to learn a score function representing the preferences in trajectories.

In other related works, Berenson et al. [5] and Phillips et al. [34] consider the problem of trajectories for high-dimensional manipulators. They store prior trajectories for computational reasons for different tasks. These methods are complementary to ours, in that we could leverage their database of trajectories and train our system on samples drawn from it. Other recent works such as [14, 13, 45] consider generating human-like trajectories. These works are complementary to ours in that humans-robot interaction is an important aspect and such ideas could be incorporated in our approach.

[3] Consider the following analogy. In search engine results, it is much harder for the user to provide the best web-pages for each query, but it is easier to provide relative ranking on the search results by clicking.

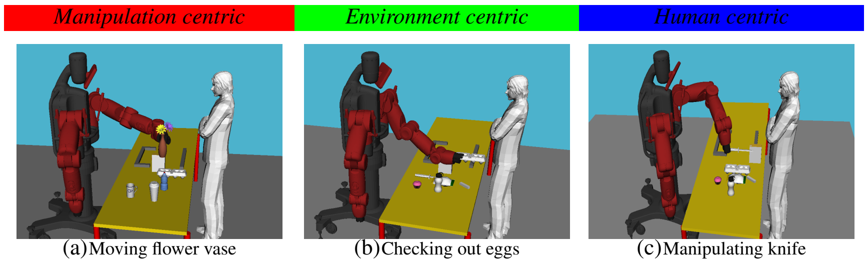
In past, learning from demonstration [18, 31] and various interactive methods (e.g. human gestures) [8, 43] have been employed to teach assembly line robots. However, these methods either required the user to demonstrate an optimal trajectory or interactively show the complete sequence of actions which the robot remembered for future use. Recent works [32, 33] in human robot collaboration learn human preferences over a sequence of sub-tasks in assembly line manufacturing. However, these works are agnostic to the user preferences over robot's trajectories. Our algorithm can complement their's by learning preferences over the trajectories thereby achieving better human robot collaboration.

6 Experiments and Results

We first describe our experimental setup, then present quantitative results (Section 6.2) , and then present robotic experiments on PR2 and Baxter (Section 6.3).

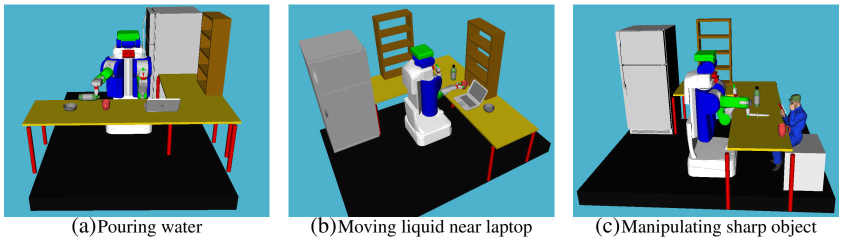
6.1 Experimental Setup

Task and Activity Set for Evaluation. We evaluate our approach on 35 robotic tasks in household setting and 16 pick-and-place tasks in a grocery store checkout setting. For household activities we use PR2, and use Baxter for the grocery store setting. To assess the generalizability of our approach, for each task we train and test on scenarios with different objects being manipulated, and/or with a different environment. We evaluate the quality of trajectories after the robot has grasped the items and while it moves them for checkout. Our work complements previous works on grasping items [40, 28], pick and place tasks [20], and detecting bar code for grocery checkout [24]. We consider following three most commonly occurring activities in household and grocery stores:
1) Manipulation centric: These activities primarily care for the object being manipulated. Hence the object's properties and the way robot moves it in the environment is more relevant. Examples of such household activities are pouring water into a cup or inserting pen inside a pen holder, Figure 6 (Left). While in grocery store such activities could include moving flower vase, or moving fruits and vegetables, which can be damaged when dropped/pushed into other items. We consider *pick-and-place, pouring and inserting activities* with following objects: *cup, bowl, bottle, pen, cereal box, flower vase, tomatoes*. Further, in every environment we place many objects, alongwith the object to be manipulated, to restrict simple straight line trajectories.
2) Environment centric: These activities also care for the interactions of the object being manipulated with the surrounding objects. Our object-object interaction features [19] allow the algorithm to learn preferences on trajectories for moving fragile objects like egg cartons or moving liquid near electronic devices, Figure 6 (Middle). We consider moving *fragile items like egg carton, heavy metal boxes near a glass table, water near laptop and other electronic devices*.
3) Human centric: Sudden movements by the robot put the human in a danger of getting hurt. We consider activities where a robot manipulates *sharp objects such as knife*, Figure 6 (Right), *moves a hot coffee cup or a bowl of water with a human in vicinity*.
Baseline algorithms. We evaluate the algorithms that learn preferences from online feedback, under two settings: (a) *untrained*, where the algorithms learn preferences



(a) Moving flower vase (b) Checking out eggs (c) Manipulating knife

Baxter in a grocery store setting.



(a) Pouring water (b) Moving liquid near laptop (c) Manipulating sharp object

PR2 in a household setting.

Fig. 6: Robot demonstrating different grocery store and household activities with various objects (**Left**) *Manipulation centric:* while pouring water the tilt angle of bottle must change in a particular manner, similarly a flower vase should be kept upright. (**Middle**) *Environment centric:* laptop is an electronic device so robot must carefully move water near it, similarly eggs are fragile and should not be lifted too high. (**Right**) *Human centric:* knife is sharp and interacts with nearby soft items and humans. It should strictly be kept at a safe distance from humans. (**Best viewed in color**)

for the new task from scratch without observing any previous feedback; (b) *pre-trained*, where the algorithms are pre-trained on other similar tasks, and then adapt to the new task. We compare the following algorithms:

- *Geometric*: It plans a path, independent of the task, using a BiRRT [27] planner.
- *Manual*: It plans a path following certain manually coded preferences.
- *TPP*: Our algorithm, evaluated under both *untrained* and *pre-trained* settings.
- *Oracle-svm*: This algorithm leverages the expert's labels on trajectories (hence the name *Oracle*) and is trained using SVM-rank [21] in a batch manner. This algorithm is *not realizable in practice*, as it requires labeling on the large space of trajectories. We use this only in pre-trained setting and during prediction it just predicts once and does not learn further.
- *MMP-online*: This is an online implementation of Maximum margin planning (MMP) [37, 38] algorithm. MMP attempts to make an expert's trajectory better than any other trajectory by a margin, and can be interpreted as a special case of our algorithm with 1-informative feedback. However, adapting MMP to our experiments poses two challenges: (i) we do not have knowledge of optimal trajectory; and (ii) the state space of the manipulator we consider is too large, and discretizing makes learning via MMP intractable. We therefore train MMP from online user feedback observed on a set of trajectories. We further treat the observed feedback as optimal. At every iteration we train a structural support vector machine (SSVM) [22] using all previous feedback as training examples, and use the learned weights to predict trajectory scores for the next iteration. Since we

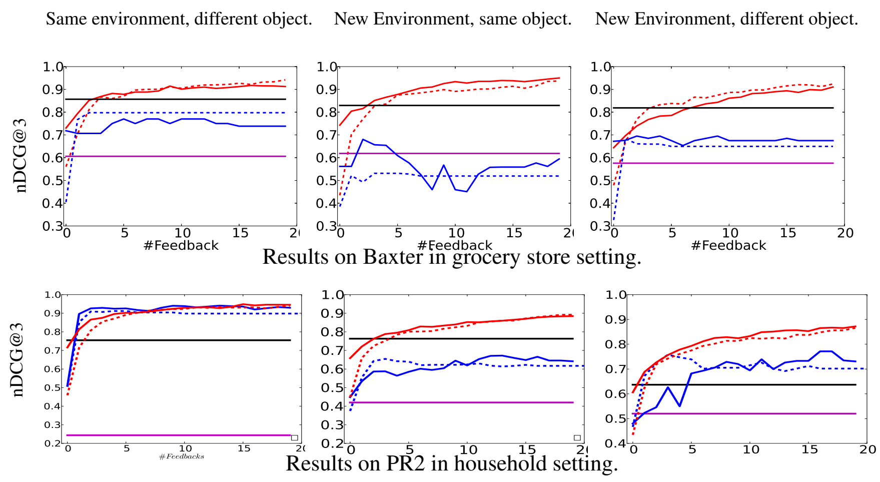


Fig. 7: Study of generalization with change in object, environment and both. Manual, Oracle-SVM, Pre-trained MMP-online (—), Untrained MMP-online (– –), Pre-trained TPP (—), Untrained TPP (– –).

learn on a set of trajectories, the argmax operation in SSVM remains tractable. We quantify closeness of trajectories by the l_2-norm of difference in their feature representations, and choose the regularization parameter C for training SSVM in hindsight, to give an unfair advantage to MMP-online.

Evaluation metrics. In addition to performing a user study (Section 6.3), we also designed two datasets to quantitatively evaluate the performance of our online algorithm. We obtained experts labels on 1300 trajectories in grocery setting and 2100 trajectories in household setting. Labels were on the basis of subjective human preferences on a Likert scale of 1-5 (where 5 is the best). Note that these absolute ratings are never provided to our algorithms and are only used for the quantitative evaluation of different algorithms. We quantify the quality of a ranked list of trajectories by its normalized discounted cumulative gain (nDCG) [30] at positions 1 and 3. While nDCG@1 is a suitable metric for autonomous robots that execute the top ranked trajectory (e.g., grocery checkout), nDCG@3 is suitable for scenarios where the robot is supervised by humans, (e.g., assembly lines). We also report average nDCG value over a given number of feedback iterations.

6.2 Results and Discussion

We now present the quantitative results where we compare TPP against the baseline algorithms on the data set of labeled trajectories.

How well does TPP generalize to new tasks? To study generalization of preference feedbacks we evaluate performance of TPP-pre-trained (i.e., *TPP* algorithm under *pre-trained* setting) on a set of tasks the algorithm has not seen before. We study generalization when: (a) only the object being manipulated changes, e.g., a bowl replaced by a cup or an egg carton replaced by tomatoes, (b) only the surrounding environment changes, e.g., rearranging objects in the environment or changing the start location of tasks, and (c) when both change. Figure 7 shows nDCG@3 plots averaged over tasks for all types of activities for both household and grocery

Algorithms	Grocery store setting on Baxter.				Household setting on PR2.			
	Manip. centric	Environ. centric	Human centric	Mean	Manip. centric	Environ. centric	Human centric	Mean
Geometric	.46 (.48)	.45 (.39)	0.31 (.30)	.40 (.39)	.36 (.54)	.43 (.38)	.36 (.27)	.38 (.40)
Manual	.61 (.62)	.77 (.77)	.33 (.31)	.57 (.57)	.53 (.55)	.39 (.53)	.40 (.37)	.44 (.48)
MMP-online	.47 (.50)	.54 (.56)	.33 (.30)	.45 (.46)	.83 (.82)	.42 (.51)	.36 (.33)	.54 (.55)
TPP	**.88 (.84)**	**.90 (.85)**	**.90 (.80)**	**.89 (.83)**	**.93 (.92)**	**.85 (.75)**	**.78 (.66)**	**.85 (.78)**

Table 1: **Comparison of different algorithms in untrained setting.** Table contains nDCG@1(nDCG@3) values averaged over 20 feedbacks.

store settings.[4] TPP-pre-trained starts-off with higher nDCG@3 values than TPP-untrained in all three cases. Further, as more feedback are provided, performance of both algorithms improves and they eventually give identical performance. We further observe, generalizing to tasks with both new environment and object is harder than when only one of them changes.

How does TPP compare to MMP-online? MMP-online proceeds by assuming every user feedback as optimal, and hence over the time it accumulates many contradictory/sub-optimal training examples. We empirically observe MMP-online generalizes better in grocery store setting than the household setting (Figure 7), however under both settings its performance remains much below TPP. This also highlights the sensitivity of MMP to sub-optimal demonstrations.

How does TPP compare to Oracle-svm? Oracle-svm starts off with nDCG@3 values higher than any other algorithm (Figure 7). The reason being, it is pre-trained using expert's labels on trajectories, and for the same reason it not realizable in practice. Furthermore, in less than 5 feedback on new task TPP improves over Oracle-svm, which is not updated since it requires expert's labels on test set.

How does TPP compare to Manual? We encode some preferences into the planners e.g., keep a glass of water upright. However, some preferences are difficult to specify, e.g., not to move heavy objects over fragile items. We empirically found (Figure 7) the resultant manual algorithm produces poor trajectories – in comparison with TPP – with an average nDCG@3 of 0.44 over all types of household activities. Table 1 reports nDCG values averaged over 20 feedback in untrained setting. For both household and grocery checkout activities TPP performs better than other baseline algorithms.

How does TPP perform with weaker feedback? To study the robustness of TPP to less informative feedback we consider the following variants of re-rank feedback:

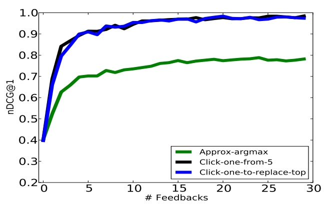


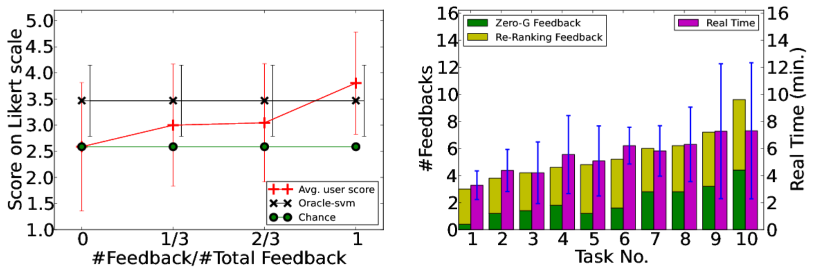
Fig. 8: Study of re-rank feedback.

1. Click-one-to-replace-top: User observes the trajectories sequentially in order of their current predicted scores and clicks on the first trajectory which is better than the top ranked trajectory.

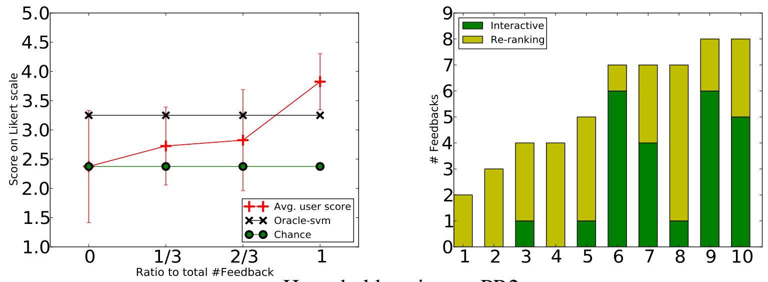
2. Click-one-from-5: Top 5 trajectories are shown and user clicks on the one he thinks is the best after watching all 5 of them.

3. Approximate-argmax: This is a weaker feedback, here instead of presenting top ranked trajectories, five random trajectories are selected as candidate. The user se-

[4] Similar results were obtained with nDCG@1 metric, not included here due to space constraints.



Grocery store setting on Baxter.



Household setting on PR2.

Fig. 9: **(Left)** Average quality of the learned trajectory after every one-third of total feedback. **(Right)** Bar chart showing the average number of feedback (re-ranking and zero-G) and time required (only for grocery store setting) for each task. Task difficulty increases from 1 to 10.

lects the best trajectory among these 5 candidates. This simulates a situation when computing an argmax over trajectories is prohibitive and therefore approximated. Figure 8 shows the performance of TPP-untrained receiving different kinds of feedback and averaged over three types of activities in grocery store setting. When feedback is more α-informative the algorithm requires fewer of those to learn preferences. In particular, click-one-to-replace-top and click-one-from-5 are more informative than approximate-argmax and therefore require fewer feedback to reach a given nDCG@1 value. Approximate-argmax being the least informative continues to show slow improvement. Since all the feedback are α-informative, for some $\alpha > 0$, eventually TPP-untrained is able to learn the preferences.

6.3 *Robotic Experiment: User Study in learning trajectories*

We perform a user study of our system on Baxter and PR2 on a variety of tasks of varying difficulties in grocery store and household settings respectively. Thereby we show our approach is practically realizable, and the combination of re-rank, zero-G/interactive feedback allows users to train the robot in few feedback.

Experiment setup: In this study, five users (not associated with this work) used our system to train Baxter on grocery checkout tasks, using zero-G and re-rank feedback. For training Baxter, the users provided zero-G feedback kinesthetically on the robot, while re-rank was elicited in a simulator. For PR2, in place of zero-G, two users provided interactive feedback on Rviz simulator. The two users were familiar

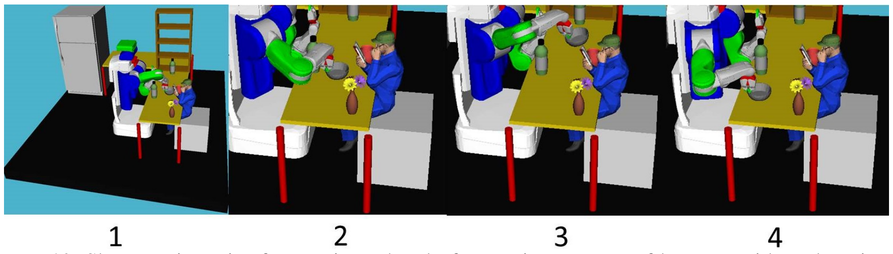


Fig. 10: Shows trajectories for moving a bowl of water in presence of human. Without learning robot plans an undesirable trajectory and moves bowl over the human (waypoints 1-3-4). After six user feedback robot learns the desirable trajectory (waypoints 1-2-4).

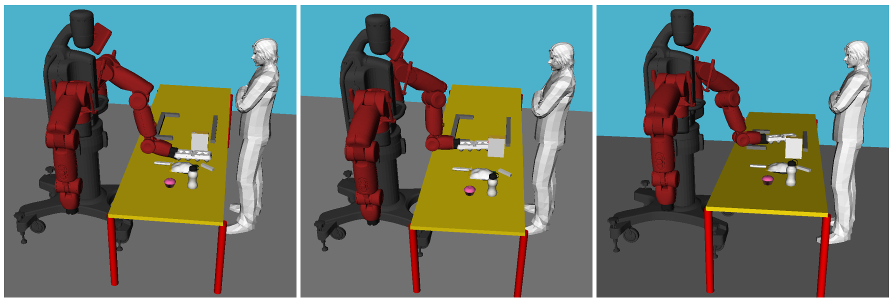


Fig. 11: Shows the learned trajectory for moving an egg carton. Since eggs are fragile robot moves the carton near the table surface. (Left) Start of trajectory. (Middle) Intermediate waypoint with egg close to the table surface. (Right) End of trajectory.

with Rviz-ROS trained PR2 on household tasks.[5] A set of 10 tasks of varying difficulty level was presented to users one at a time, and they were instructed to provide feedback until they were satisfied with the top ranked trajectory. To quantify the quality of learning each user evaluated their own trajectories (self score), the trajectories learned by the other users (cross score), and those predicted by Oracle-svm, on a Likert scale of 1-5 (where 5 is the best). We also recorded the time a user took for each task – from start of training till the user was satisfied with the top ranked trajectory.

Is re-rank feedback easier to elicit from users than zero-G or interactive? In our user study, on average a user took 3 re-rank and 2 zero-G feedback per task to train a robot (Table 2). From this we conjecture, for high DoF manipulators re-rank feedback is easier to provide than zero-G – which requires modifying the manipulator joint angles. However, an increase in the number of zero-G (interactive) feedback with task difficulty suggests, Figure 9 (Right), users rely more on zero-G feedback for difficult tasks since it allows precisely rectifying erroneous waypoints. Figure 10 and Figure 11 show two example trajectories learned by a user.

[5] The smaller user size on PR2 is because it requires users with experience in Rviz-ROS. Further, we also observed users found it harder to correct trajectory waypoints in a simulator than providing zero-G feedback on the robot. For the same reason we report training time only on Baxter for grocery store setting.

User	# Re-ranking feedback	# Zero-G feedback	Average time (min.)	Trajectory-Quality self	Trajectory-Quality cross
1	5.4 (4.1)	3.3 (3.4)	7.8 (4.9)	3.8 (0.6)	4.0 (1.4)
2	1.8 (1.0)	1.7 (1.3)	4.6 (1.7)	4.3 (1.2)	3.6 (1.2)
3	2.9 (0.8)	2.0 (2.0)	5.0 (2.9)	4.4 (0.7)	3.2 (1.2)
4	3.2 (2.0)	1.5 (0.9)	5.3 (1.9)	3.0 (1.2)	3.7 (1.0)
5	3.6 (1.0)	1.9 (2.1)	5.0 (2.3)	3.5 (1.3)	3.3 (0.6)

User	# Re-ranking feedbacks	# Interactive feedbacks	Trajectory-Quality self	Trajectory-Quality cross
1	3.1 (1.3)	2.4 (2.4)	3.5 (1.1)	3.6 (0.8)
2	2.3 (1.1)	1.8 (2.7)	4.1 (0.7)	4.1 (0.5)

Table 2: Shows learning statistics for each user. Self and cross scores of the final learned trajectories. The number inside bracket is standard deviation. **(Top)** Results for grocery store on Baxter. **(Bottom)** Household setting on PR2.

How many feedback a user takes to improve over Oracle-svm? On average, a user took 5 feedback to improve over Oracle-svm, Figure 9 (Left), which is also consistent with our quantitative analysis. In grocery setting, user 4 and 5 were critical towards trajectories learned by oracle-svm and gave them low scores. This indicate a possible mismatch in preferences between our expert (on whose labels oracle-svm trained) and user 4, 5.
How do users' unobserved score functions vary? An average difference of 0.6 between users' self and cross score (Table 2) in grocery checkout setting suggests preferences varied across users, but only marginally. In situations where this difference is significant and a system is desired for a user population, a future work might explore coactive learning for satisfying user population which has recently been applied to search engines [35]. For household setting the sample size is small to draw a such conclusion.
How long does it take for users to train a robot? We report training time only for grocery store setting, because the interactive feedback in household setting requires users with experience in Rviz-ROS. Further, we observed that users found it difficult to modify robot's joint angles in a simulator to their desired configuration. In grocery checkout setting, among all the users, user 1 had the strictest preferences and also experienced some early difficulties in using the system and therefore took longer than others. On an average, a user took 5.5 minutes per task which we believe is acceptable for most applications. Future research in human computer interaction, visualization and better user interface [42] could further reduce this time. For example, simultaneous visualization of top ranked trajectories instead of sequentially showing them to users (which we currently do) could bring down the time for rerank feedback. Despite its limited size, through user study we show our algorithm is realizable in practice on high DoF manipulators. We hope this motivates researchers to build robotic systems capable of learning from *non-expert* users.
For more details, videos & code, visit: `http://pr.cs.cornell.edu/coactive/`

7 Conclusion

With manipulators in human environments, it is important for robots to plan motions that follow user's preferences. In this work, we considered preferences that go beyond simple geometric constraints and that considered surrounding context of various objects and humans in the environment. We presented a coactive learning approach for training robots these preferences through iterative improvements from non-expert users. Unlike in standard learning from demonstration approaches, our approach does not require the user to provide optimal trajectories as training data. We evaluated our approach on various household (with PR2) and grocery store checkout settings (with Baxter). Our experiments suggest that it is indeed possible to train robots within a few minutes with just a few incremental feedbacks from non-expert users.

Acknowledgments. This research was supported by ARO, Microsoft Faculty fellowship and NSF Career award (to Saxena).

References

[1] P. Abbeel, A. Coates, and A. Y. Ng. Autonomous helicopter aerobatics through apprenticeship learning. *IJRR*, 29(13), 2010.
[2] B. Akgun, M. Cakmak, K. Jiang, and A. L. Thomaz. Keyframe-based learning from demonstration. *IJSR*, 4(4):343–355, 2012.
[3] R. Alterovitz, T. Siméon, and K. Goldberg. The stochastic motion roadmap: A sampling framework for planning with markov motion uncertainty. In *RSS*, 2007.
[4] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009.
[5] D. Berenson, P. Abbeel, and K. Goldberg. A robot path planning framework that learns from experience. In *ICRA*, 2012.
[6] J. V. D. Berg, P. Abbeel, and K. Goldberg. Lqg-mp: Optimized path planning for robots with motion uncertainty and imperfect state information. In *RSS*, 2010.
[7] S. Bhattacharya, M. Likhachev, and V. Kumar. Identification and representation of homotopy classes of trajectories for search-based path planning in 3d. In *RSS*, 2011.
[8] R. Bischoff, A. Kazi, and M. Seyfarth. The morpha style guide for icon-based programming. In *Proceedings. 11th IEEE International Workshop on RHIC.*, 2002.
[9] S. Calinon, F. Guenter, and A. Billard. On learning, representing, and generalizing a task in a humanoid robot. *IEEE Transactions on Systems, Man, and Cybernetics*, 2007.
[10] B. J. Cohen, S. Chitta, and M. Likhachev. Search-based planning for manipulation with motion primitives. In *ICRA*, 2010.
[11] D. Dey, T. Y. Liu, M. Hebert, and J. A. Bagnell. Contextual sequence prediction with application to control library optimization. In *RSS*, 2012.
[12] R. Diankov. *Automated Construction of Robotic Manipulation Programs*. PhD thesis, CMU, RI, 2010.
[13] A. Dragan and S. Srinivasa. Generating legible motion. In *RSS*, 2013.
[14] A. Dragan, K. Lee, and S. Srinivasa. Legibility and predictability of robot motion. In *HRI*, 2013.
[15] L. H. Erickson and S. M. LaValle. Survivability: Measuring and ensuring path diversity. In *ICRA*, 2009.
[16] D. Gossow, A. Leeperand D. Hershberger, and M. Ciocarlie. Interactive markers: 3-d user interfaces for ros applications [ros topics]. *Robotics & Automation Magazine, IEEE*, 18(4): 14–15, 2011.
[17] C. J. Green and A. Kelly. Toward optimal sampling in the space of paths. In *ISRR*. 2007.
[18] G. E. Hovland, P. Sikka, and B. J. McCarragher. Skill acquisition from human demonstration using a hidden markov model. In *ICRA*, 1996.
[19] A. Jain, B. Wojcik, T. Joachims, and A. Saxena. Learning trajectory preferences for manipulators via iterative improvement. In *NIPS*, 2013.
[20] Y. Jiang, M. Lim, C. Zheng, and A. Saxena. Learning to place new objects in a scene. *IJRR*, 31(9), 2012.

[21] T. Joachims. Training linear svms in linear time. In *KDD*, 2006.
[22] T. Joachims, T. Finley, and C. Yu. Cutting-plane training of structural svms. *Mach Learn*, 77 (1), 2009.
[23] S. Karaman and E. Frazzoli. Incremental sampling-based algorithms for optimal motion planning. In *RSS*, 2010.
[24] E. Klingbeil, D. Rao, B. Carpenter, V. Ganapathi, A. Y. Ng, and O. Khatib. Grasping with application to an autonomous checkout robot. In *ICRA*, 2011.
[25] J. Kober and J. Peters. Policy search for motor primitives in robotics. *Machine Learning*, 84 (1), 2011.
[26] H. S. Koppula and A. Saxena. Anticipating human activities using object affordances for reactive robotic response. In *RSS*, 2013.
[27] S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. *IJRR*, 20(5):378–400, 2001.
[28] I. Lenz, H. Lee, and A. Saxena. Deep learning for detecting robotic grasps. In *RSS*, 2013.
[29] S. Levine and V. Koltun. Continuous inverse optimal control with locally optimal examples. In *ICML*, 2012.
[30] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*, volume 1. Cambridge University Press Cambridge, 2008.
[31] M. N. Nicolescu and M. J. Mataric. Natural methods for robot task learning: Instructive demonstrations, generalization and practice. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, 2003.
[32] S. Nikolaidis and J. Shah. Human-robot teaming using shared mental models. In *HRI, Workshop on Human-Agent-Robot Teamwork*, 2012.
[33] S. Nikolaidis and J. Shah. Human-robot cross-training: Computational formulation, modeling and evaluation of a human team training strategy. In *IEEE/ACM ICHRI*, 2013.
[34] M. Phillips, B. Cohen, S. Chitta, and M. Likhachev. E-graphs: Bootstrapping planning with experience graphs. In *RSS*, 2012.
[35] K. Raman and T. Joachims. Learning socially optimal information systems from egoistic users. In *Proc. ECML*, 2013.
[36] N. Ratliff. *Learning to Search: Structured Prediction Techniques for Imitation Learning*. PhD thesis, CMU, RI, 2009.
[37] N. Ratliff, J. A. Bagnell, and M. Zinkevich. Maximum margin planning. In *ICML*, 2006.
[38] N. Ratliff, D. Silver, and J. A. Bagnell. Learning to search: Functional gradient techniques for imitation learning. *Autonomous Robots*, 27(1):25–53, 2009.
[39] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa. Chomp: Gradient optimization techniques for efficient motion planning. In *ICRA*, 2009.
[40] A. Saxena, J. Driemeyer, and A.Y. Ng. Robotic grasping of novel objects using vision. *IJRR*, 27(2), 2008.
[41] P. Shivaswamy and T. Joachims. Online structured prediction via coactive learning. In *ICML*, 2012.
[42] B. Shneiderman and C. Plaisant. *Designing The User Interface: Strategies for Effective Human-Computer Interaction*. Addison-Wesley Publication, 2010.
[43] A. Stopp, S. Horstmann, S. Kristensen, and F. Lohnert. Towards interactive learning for manufacturing assistants. In *Proceedings. 10th IEEE International Workshop on RHIC.*, 2001.
[44] I. A. Sucan, M. Moll, and L. E. Kavraki. The Open Motion Planning Library. *IEEE Robotics & Automation Magazine*, 19(4):72–82, 2012. `http://ompl.kavrakilab.org`.
[45] K. Tamane, M. Revfi, and T. Asfour. Synthesizing object receiving motions of humanoid robots with human motion database. In *ICRA*, 2013.
[46] P. Vernaza and J. A. Bagnell. Efficient high dimensional maximum entropy modeling via symmetric partition functions. In *NIPS*, 2012.
[47] A. Wilson, A. Fern, and P. Tadepalli. A bayesian approach for policy learning from trajectory preference queries. In *NIPS*, 2012.
[48] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey. Maximum entropy inverse reinforcement learning. In *AAAI*, 2008.