Learning preferences for manipulation tasks from online coactive feedback

Ashesh Jain, Shikhar Sharma, Thorsten Joachims 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 simple geometric constraints on trajectories; they are rather governed by the surrounding context of various objects and human interactions in the environment. We propose a coactive online learning framework for teaching preferences in contextually rich environments. The key novelty of our approach lies in the type of feedback expected from the user: the human user does not need to demonstrate optimal trajectories as training data, but merely needs to iteratively provide trajectories that slightly improve over the trajectory currently proposed by the system. We argue that this coactive preference feedback can be more easily elicited than demonstrations of optimal trajectories. Nevertheless, theoretical regret bounds of our algorithm match the asymptotic rates of optimal trajectory algorithms. 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).

Keywords
Manipulation planning, online learning, user feedback

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 repetitive motions (e.g. recent deployment of Baxter on assembly lines). One challenge in the deployment of these robots in unstructured environments (such as a grocery checkout counter or at 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 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 criteria that most path planners focus on), but it also needs to satisfy the user's preferences. Such users' preferences over trajectories can be common across users or they may 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 knife 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 preferred. 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 them in existing path planners (Schulman et al., 2013; Sucan et al., 2012; Zucker et al., 2013) a priori.

In this work we propose an algorithm for learning user preferences over trajectories through interactive feedback from the user in a coactive learning setting (Shivaswamy

---

Department of Computer Science, Cornell University, USA

Corresponding author:
Ashesh Jain, Department of Computer Science, Cornell University, Gates Hall, Ithaca, NY 14850, USA.
Email: asheshjain399@gmail.com

and Joachims, 2012). In this setting the robot learns through iterations of user feedback. At each iteration robot receives a task and it predicts a trajectory. The user responds by slightly improving the trajectory but not necessarily revealing the optimal trajectory. The robot use this feedback from user to improve its predictions for future iterations. Unlike in other learning settings, where a human first demonstrates optimal trajectories for a task to the robot (Argall et al., 2009), 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 robot's trajectories, thereby learning preferences of user and not the expert. From these interactive improvements the robot learns a general model of the user's preferences in an online fashion. We realize this learning algorithm on PR2 and Baxter robots, and also leverage robot-specific design to allow users to easily give preference feedback.

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 demonstration algorithms (Ratliff et al., 2007).

In our empirical evaluation we learn preferences for two high-DoF robots, PR2 and Baxter, on a variety of household and grocery checkout tasks, respectively. We design expressive trajectory features and 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 a set of 35 household and 16 grocery checkout tasks, both in batch experiments as well as through robotic experiments wherein users provide their preferences to 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. In summary, our key contributions are as follows.

1. We present an approach for teaching robots which does not rely on experts' demonstrations but nevertheless gives strong theoretical guarantees.
2. We design a robotic system with multiple easy to elicit feedback mechanisms to improve the current trajectory.
3. We implement our algorithm on two robotic platforms, PR2 and Baxter, and support it with a user study.
4. We consider preferences that go beyond simple geometric criteria to capture object and human interactions.
5. We design expressive trajectory features to capture contextual information. These features might also find use in other robotic applications.

In the following section we discuss related works. In Section 3 we describe our system and feedback mechanisms. Our learning algorithm and trajectory features are discussed in Sections 4 and 4.3, respectively. Section 6 gives our experiments and results. We discuss future research directions and conclude in Section 7.

2. Related work

Path planning is one of the key problems in robotics. Here, the objective is to find a collision free path from a start to goal location. Over the last decade many planning algorithms have been proposed, such as sampling based planners by LaValle and Kuffner (2001) and Karaman and Frazzoli (2010), search based planners by Cohen et al. (2010), trajectory optimizers by Schulman et al. (2013), and Zucker et al. (2013) and many more (Karaman and Frazzoli, 2011). However, given the large space of possible trajectories in most robotic applications simply a collision-free trajectory might not suffice, instead the trajectory should satisfy certain constraints and obey the end user preferences. Such preferences are often encoded as a cost which planners optimize (Karaman and Frazzoli, 2010; Schulman et al., 2013; Zucker et al., 2013). We address the problem of learning a cost over trajectories for context-rich environments, and from sub-optimal feedback elicited from non-expert users. We now describe related work in various aspects of this problem.

2.1. Learning from Demonstration

Teaching a robot to produce desired motions has been a long standing goal and several approaches have been studied. In Learning from Demonstration (LfD) an expert provides demonstrations of optimal trajectories and the robot tries to mimic the expert. Examples of LfD includes, autonomous helicopter flights (Abbeel et al., 2010), ball-in-a-cup game (Kober and Peters, 2011), planning two-dimensional paths (Ratliff et al., 2006; Ratliff, 2009), etc. Such settings 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 (Akgun et al., 2012).[1] 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 Ratliff et al. (2007) the robot observes optimal user feedback but performs approximate inference. On the other hand, in our setting, the user never discloses the optimal trajectory or feedback, but instead, the robot learns preferences from sub-optimal suggestions for how the trajectory can be improved.

2.2. Noisy demonstrations and other forms of user feedback

Some later works in LfD provided ways for handling noisy demonstrations, under the assumption that demonstrations

are either near optimal, as in Ziebart et al. (2008), or locally optimal, as in Levine and Koltun (2012). 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. Wilson et al. (2012) proposed a Bayesian framework for learning rewards of a Markov decision process via trajectory preference queries. Our approach advances over those of Wilson et al. (2012) and Calinon et al. (2007) in that we model user as a utility maximizing agent. Further, our score function theoretically converges to user's hidden function despite receiving sub-optimal feedback. In the past, various interactive methods (e.g. human gestures) (Stopp et al., 2001; Bischoff et al., 2002) have been employed to teach assembly line robots. However, these methods required the user to interactively show the complete sequence of actions, which the robot then remembered for future use. Recent works by Nikolaidis and Shah (2012, 2013) in human–robot collaboration learns 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 work could complement theirs to achieve better human–robot collaboration.

2.3. Learning preferences over trajectories

User preferences over robot's trajectories have been studied in human–robot interaction. Sisbot et al. (2007b,a) and Mainprice et al. (2011) planned trajectories satisfying user-specified preferences in the form of constraints on the distance of the robot from the user, the visibility of the robot and the user's arm comfort. Dragan and Srinivasa (2013) used functional gradients (Ratliff et al., 2009b) to optimize for legibility of robot trajectories. We differ from these in that we take a data-driven approach and *learn* score functions reflecting user preferences from sub-optimal feedback.

2.4. Planning from a cost function

In many applications, the goal is to find a trajectory that optimizes a cost function. Several works build upon the sampling-based planner RRT (LaValle and Kuffner, 2001) to optimize various cost heuristics (Ettlin and Bleuler, 2006; D. Ferguson and Stentz, 2006; Jaillet et al., 2010). Additive cost functions with Lipschitz continuity can be optimized using optimal planners such as RRT* (Karaman and Frazzoli, 2010). Some approaches introduce sampling bias (Leven and Hutchinson, 2003) to guide the sampling-based planner. Recent trajectory optimizers such as CHOMP (Ratliff et al., 2009b) and TrajOpt (Schulman et al., 2013) provide optimization based approaches to finding optimal trajectory. Our work is complementary to these in that we learn a cost function while the above approaches optimize a cost.

Our work is also complementary to few works in path planning. Berenson et al. (2012) and Phillips et al. (2012) consider the problem of trajectories for high-dimensional manipulators. For computational reasons they create a database of prior trajectories, which we could leverage to train our system. Other recent works consider generating human-like trajectories (Dragan et al., 2013; Dragan and Srinivasa, 2013; Tamane et al., 2013). Human–robot interaction is an important aspect and our approach could incorporate similar ideas.

2.5. Application domain

In addition to above-mentioned differences we also differ in the applications we address. We capture the necessary contextual information for household and grocery store robots, while such context is absent in previous works. 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. Some works in mobile robotics learn context-rich perception-driven cost functions, such as Silver et al. (2010), Kretzschmar et al. (2014) and Kitani et al. (2012). In this work we use features that consider robot configurations, object–object relations, and temporal behavior, and use them to learn a score function representing the preferences over trajectories.

3. Coactive learning with incremental feedback

We first give an overview of our robot learning setup and then describe in detail three mechanisms of user feedback.

3.1. Robot learning setup

We propose an online algorithm for learning preferences in trajectories from sub-optimal user feedback. At each step the robot receives a task as input and outputs a trajectory that maximizes its current estimate of some score function. It then observes 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. We implement the algorithm on PR2 and Baxter robots, both having two 7-DoF arms. In the process of training, the initial trajectory proposed by the robot can be far-off the desired behavior. Therefore, instead of directly executing trajectories in human environments, users first visualize them in the OpenRAVE simulator (Diankov, 2010) and then decide the kind of feedback they would like to provide.

3.2. Feedback mechanisms

Our goal is to learn even from feedback given by non-expert users. We therefore require the feedback only to be *incrementally* better (as compared with being close to

Fig. 1. Zero-G feedback: learning trajectory preferences from suboptimal *zero-G* feedback. (Left) Robot plans a bad trajectory (waypoints 1–2–4) with knife close to flower. As feedback, user corrects waypoint 2 and moves it to waypoint 3. (Right) User providing *zero-G* feedback on waypoint 2.

Fig. 2. Re-rank feedback mechanism. (Left) Robot ranks trajectories using the score function. (Middle) Top three trajectories on a touch screen device (iPad here). (Right) As feedback, the user improves the ranking by selecting the third trajectory.

optimal) in expectation, and will show that such feedback is sufficient for the algorithm's convergence. This stands in contrast to LfD methods (Ratliff et al., 2006; Ratliff, 2009; Abbeel et al., 2010; Kober and Peters, 2011) which require (near-)optimal demonstrations of the complete trajectory. Such demonstrations can be extremely challenging and non-intuitive to provide for many high-DoF manipulators (Akgun et al., 2012). Instead, we found (Jain et al., 2013b,a) that it is more intuitive for users to give incremental feedback on high-DoF arms by improving upon a proposed trajectory. We now summarize three feedback mechanisms that enable the user to iteratively provide improved trajectories.

3.2.1. Re-ranking. We display the ranking of trajectories using OpenRAVE (Diankov, 2010) 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. The 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 2 shows three trajectories for moving a knife. As feedback, the 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 (1st), green (2nd) and blue (3rd). Since eggs are fragile the user selects the green trajectory.
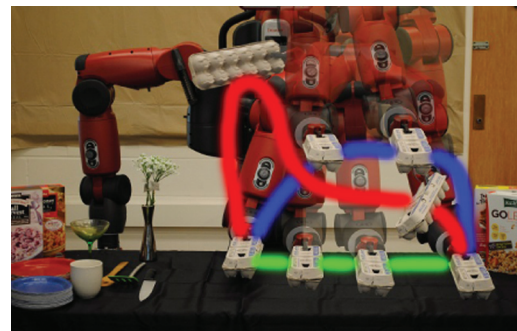

Fig. 3. Re-ranking feedback: three trajectories for moving egg carton from left to right. Using the current estimate of score function robot ranks them as red, green and blue. As feedback user clicks the green trajectory. Preference: Eggs are fragile. They should be kept upright and near the supporting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

3.2.2. Zero-G. This is a kinesthetic feedback. It allows the user to correct trajectory waypoints by physically changing the robot's arm configuration as shown in Figure 1. High-DoF arms such as the Barrett WAM and Baxter have zero-force gravity-compensation (zero-G) mode, under which the robot's arms become light and the users can effortlessly steer them to a desired configuration. On Baxter, this zero-G mode is automatically activated when a user holds the robot's wrist (see Figure 1, right). We use this zero-G mode
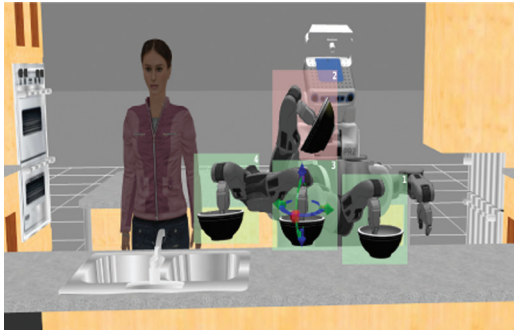
Fig. 4. Interactive feedback. The task here is to move a bowl filled with water. The robot presents a bad trajectory with waypoints 1–2–4 to the user. As feedback user moves waypoint 2 (red) to waypoint 3 (green) using Rviz interactive markers. The interactive markers guide the user to correct the waypoint.

as a feedback method for incrementally improving the trajectory by correcting a waypoint. 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.

3.2.3. Interactive. For the robots whose hardware does not permit zero-G feedback, such as PR2, we built an alternative interactive Rviz-ROS (Gossow et al., 2011) 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.

Note that, in all three kinds of feedback the user never reveals the optimal trajectory to the algorithm, but only provides a slightly improved trajectory (in expectation).

4. 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 they value 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.

4.1. 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.

4.2. 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 $s^*(x, y)$, $y_t^* = \arg\max_y s^*(x_t, y)$. 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.

4.3. Expert versus non-expert user

We refer to an expert user as someone who can demonstrate the optimal trajectory y^* to the robot. For example, robotics experts such as, the pilot demonstrating helicopter maneuver in Abbeel et al. (2010). On the other hand, our non-expert users never demonstrate y^*. They can only provide feedback $\bar{y}$ indicating $s^*(x, \bar{y}) > s^*(x, y)$. For example, users working with assistive robots on assembly lines.

5. Learning algorithm

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

$$s(x, y; w) = w \cdot \phi(x, y) \qquad (1)$$

where w is a weight vector that needs to be learned, and $\phi(\cdot)$ are features describing trajectory y for context x. Such linear representation of score functions have been previously used for generating desired robot behaviors (Ratliff et al., 2006; Ziebart et al., 2008; Abbeel et al., 2010).

We further decompose the score function in two parts, one only concerned with the objects the trajectory is
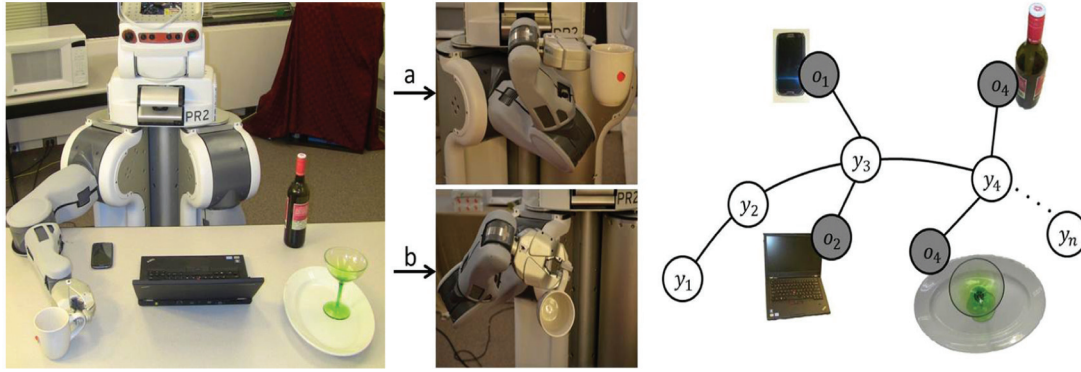
Fig. 5. (Left) An environment with a few objects where the robot was asked to move the cup on the left to the right. (Middle) There are two ways of moving it, 'a' and 'b', both are suboptimal in that the arm is contorted in 'a' but it tilts the cup in 'b'. Given such constrained scenarios, we need to reason about such subtle preferences. (Right) We encode preferences concerned with object–object interactions in a score function expressed over a graph. Here $y_1, …, y_n$ are different waypoints in a trajectory. The shaded nodes corresponds to environment (table node not shown here). Edges denotes interaction between nodes.

interacting with, and the other with the object being manipulated and the environment

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

We now describe the features for the two terms, $\phi_O(\cdot)$ and $\phi_E(\cdot)$ in the following.

5.1. Features describing object–object interactions

This feature captures the interaction between objects in the environment with the object being manipulated. We enumerate waypoints of trajectory y as $y_1,…,y_N$ and objects in the environment as $\mathcal{O} = \{o_1, …, o_K\}$. The robot manipulates the object $\bar{o} \in \mathcal{O}$. A few of the trajectory waypoints would be affected by the other objects in the environment. For example in Figure 5, o_1 and o_2 affect the waypoint y_3 because of proximity. Specifically, we connect an object o_k to a trajectory waypoint if the minimum distance to collision is less than a threshold or if o_k lies below $\bar{o}$. The edge connecting y_j and o_k is denoted as $(y_j, o_k) \in \mathcal{E}$.

Since it is the attributes (Koppula et al., 2011) of the object that really matter in determining the trajectory quality, we represent each object with its *attributes*. Specifically, for every object o_k, we consider a vector of M binary variables $[l_k^1, …, l_k^M]$, with each $l_k^m = \{0, 1\}$ indicating whether object o_k possesses property m or not. For example, if the set of possible properties are {heavy, fragile, sharp, hot, liquid, electronic}, then a laptop and a glass table can have labels [0, 1, 0, 0, 0, 1] and [0, 1, 0, 0, 0, 0], respectively. The binary variables l_k^p and l^q indicates whether o_k and $\bar{o}$ possess property p and q respectively.[2] Then, for every (y_j, o_k) edge, we extract following four features $\phi_{oo}(y_j, o_k)$: projection of minimum distance to

collision along x, y and z (vertical) axis and a binary variable, that is 1 if o_k lies vertically below $\bar{o}$ and 0 otherwise.

We now define the score $s_O(\cdot)$ over this graph as follows:

$$s_O(x, y; w_O) = \sum_{(y_j, o_k) \in \mathcal{E}} \sum_{p, q = 1}^{M} l_k^p l_k^q [w_{pq} \cdot \phi_{oo}(y_j, o_k)] \quad (3)$$

Here, the weight vector w_{pq} captures interaction between objects with properties p and q. We obtain w_O in (2) by concatenating vectors w_{pq}. More formally, if the vector at position i of w_O is w_{uv} then the vector corresponding to position i of $\phi_O(x, y)$ will be $\sum_{(y_j, o_k) \in \mathcal{E}} l_k^u l^v [\phi_{oo}(y_j, o_k)]$.

5.2. Trajectory features

We now describe features, $\phi_E(x, y)$, obtained by performing operations on a set of waypoints. They comprise the following three types of the features.

5.2.1. Robot arm configurations.
While a robot can reach the same operational space configuration for its wrist with different configurations of the arm, not all of them are preferred (Zacharias et al., 2011). For example, the contorted way of holding the cup shown in Figure 5 may be fine at that time instant, but would present problems if our goal is to perform an activity with it, e.g. doing the pouring activity. Furthermore, humans like to anticipate robots move and to gain users' confidence, robot should produce predictable and legible robotic motions (Dragan and Srinivasa, 2013).

We compute features capturing robot's arm configuration using the location of its elbow and wrist, with respect to its shoulder, in cylindrical coordinate system, (r, θ, z). We divide a trajectory into three parts in time and compute nine features for each of the parts. These features encode the maximum and minimum r, θ and z values for wrist and
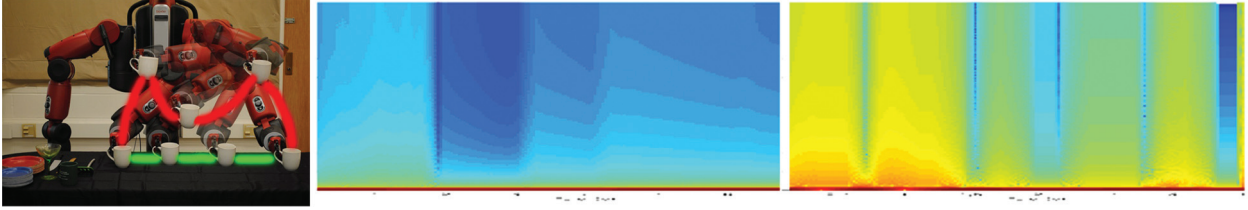
Fig. 6. (Top) A good (green) and bad (red) trajectory for moving a mug. The bad trajectory undergoes ups-and-downs. (Bottom) Spectrograms for movement in the *z*-direction: on the left is a good (green) trajectory, on the right is a bad (red) trajectory.

elbow in that part of the trajectory, giving us six features. Since at the limits of the manipulator configuration, joint locks may happen, therefore we also add 3 features for the location of robot's elbow whenever the end-effector attains its maximum r, θ and z values, respectively. Thus, obtaining $\phi_{robot}(\cdot) \in \mathbb{R}^9$ (3 + 3 + 3 = 9) features for each one-third part and $\phi_{robot}(\cdot) \in \mathbb{R}^{27}$ for the complete trajectory.

5.2.2. Orientation and temporal behaviour of the object to be manipulated.
Object orientation during the trajectory is crucial in deciding its quality. For some tasks, the orientation must be strictly maintained (e.g. moving a cup full of coffee); and for some others, it may be necessary to change it in a particular fashion (e.g. pouring activity). Different parts of the trajectory may have different requirements over time. For example, in the placing task, we may need to bring the object closer to obstacles and be more careful.

We therefore divide trajectory into three parts in time. For each part we store the cosine of the object's maximum deviation, along the vertical axis, from its final orientation at the goal location. To capture object's oscillation along trajectory, we obtain a spectrogram for each one-third part for the movement of the object in *x*, *y*, *z* directions as well as for the deviation along vertical axis (e.g. Figure 6). We then compute the average power spectral density in the low- and high-frequency part as eight additional features for each. This gives us 9 (= 1 + 4 * 2) features for each one-third part. Together with one additional feature of object's maximum deviation along the whole trajectory, we get $\phi_{obj}(\cdot) \in \mathbb{R}^{28}$ (= 9 * 3 + 1).

5.2.3. Object–environment interactions.
This feature captures temporal variation of vertical and horizontal distances of the object $\bar{o}$ from its surrounding surfaces. In detail, we divide the trajectory into three equal parts, and for each part we compute object's: (i) minimum vertical distance from the nearest surface below it; (ii) minimum horizontal distance from the surrounding surfaces; (iii) minimum distance from the table, on which the task is being performed; and (iv) minimum distance from the goal location. We also take an average, over all of the waypoints, of the horizontal and vertical distances between the object and the nearest surfaces around it.[3] To capture temporal variation of

object's distance from its surrounding we plot a time–frequency spectrogram of the object's vertical distance from the nearest surface below it, from which we extract six features by dividing it into grids. This feature is expressive enough to differentiate whether an object just grazes over table's edge (steep change in vertical distance) versus, it first goes up and over the table and then moves down (relatively smoother change). Thus, the features obtained from object–environment interaction are $\phi_{obj-env}(\cdot) \in \mathbb{R}^{20}$ (3 * 4 + 2 + 6 = 20).

Final feature vector is obtained by concatenating $\phi_{obj-env}$, ϕ_{obj} and ϕ_{robot}, giving us $\phi_E(\cdot) \in \mathbb{R}^{75}$.

5.3. 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 $s(x, y; w_O, w_E)$:

$$y^* = \arg\max_y s(x, y; w_O, w_E) \qquad (4)$$

Second, for a given set $\{y^{(1)}, \ldots, y^{(n)}\}$ of discrete trajectories, we need to compute (4). 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 are either discretizing the state space (Alterovitz et al., 2007; Bhattacharya et al., 2011; Vernaza and Bagnell, 2012) or directly sampling trajectories from the continuous space (Berg et al., 2010; Dey et al., 2012). Previously, both approaches have been studied. However, for high-DoF manipulators the sampling based approach (Berg et al., 2010; Dey et al., 2012) maintains tractability of the problem, hence we take this approach. More precisely, similar to Berg et al. (2010), we sample trajectories using rapidly-exploring random trees (RRT) (LaValle and Kuffner, 2001).[4] However, naively sampling trajectories could return many similar trajectories. To get diverse samples of trajectories we use various diversity introducing methods. For example, we introduce obstacles in the environment which forces the planner to sample different trajectories. Our methods also introduce randomness in planning by initializing goal-sample bias of RRT planner randomly. To avoid sampling similar trajectories multiple times, one of our diversity method introduce
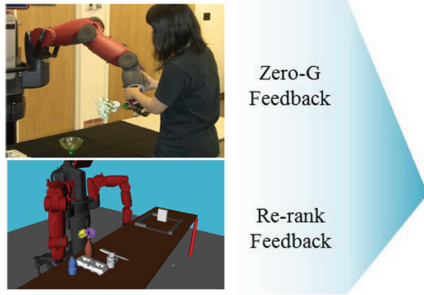
Algorithm 1. Trajectory Preference Perceptron (TPP).

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. 7. 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 **interactive** feedback in the 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 five shown on the simulator.

obstacles to block waypoints of already sampled trajectories. Recent work by Ross et al. (2013) propose the use of sub-modularity to achieve diversity. For more details on sampling trajectories we refer interested readers to the work by Erickson and LaValle (2009) and Green and Kelly (2011). Since our primary goal is to learn a score function on trajectories we now describe our learning algorithm.

5.4. Learning the scoring function

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 (Shivaswamy and Joachims, 2012) 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. Our update equation resembles to the weights update equation of Ratliff et al. (2006). However, our update does not depend on the availability of optimal demonstrations. Figure 7 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* (Shivaswamy and Joachims, 2012). 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 $\alpha \in (0, 1]$ of the maximum possible range $s^*(x_t, \bar{y}_t) - s^*(x_t, y_t)$. It is important to note that this condition only needs to be met in expectation and not deterministically. This leaves room for noisy and

imperfect user feedback. 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 (2012) can be adapted (for proof see Appendices A and B) to show that average regret of TPP is upper bounded by

$$E[REG_T] \leq \mathcal{O}\left(\frac{1}{\alpha\sqrt{T}} + \frac{1}{\alpha T}\sum_{t=1}^{T}\xi_t\right)$$

In practice, over feedback iterations the quality of trajectory y proposed by robot improves. The α-informative criterion only requires the user to improve y to $\bar{y}$ in expectation.

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 6.4).

6.1. Experimental setup

6.1.1. Task and activity set for evaluation. We evaluate our approach on 35 robotic tasks in a household setting and 16 pick-and-place tasks in a grocery store checkout setting. For household activities we use PR2, and for the grocery store setting we use Baxter. 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 item in question and while the robot moves it for task completion. Our work complements previous works on grasping items (Saxena et al., 2008; Lenz et al., 2013), pick and place tasks (Jiang et al., 2012), and detecting bar codes for grocery checkout (Klingbeil et al., 2011). We consider the following three
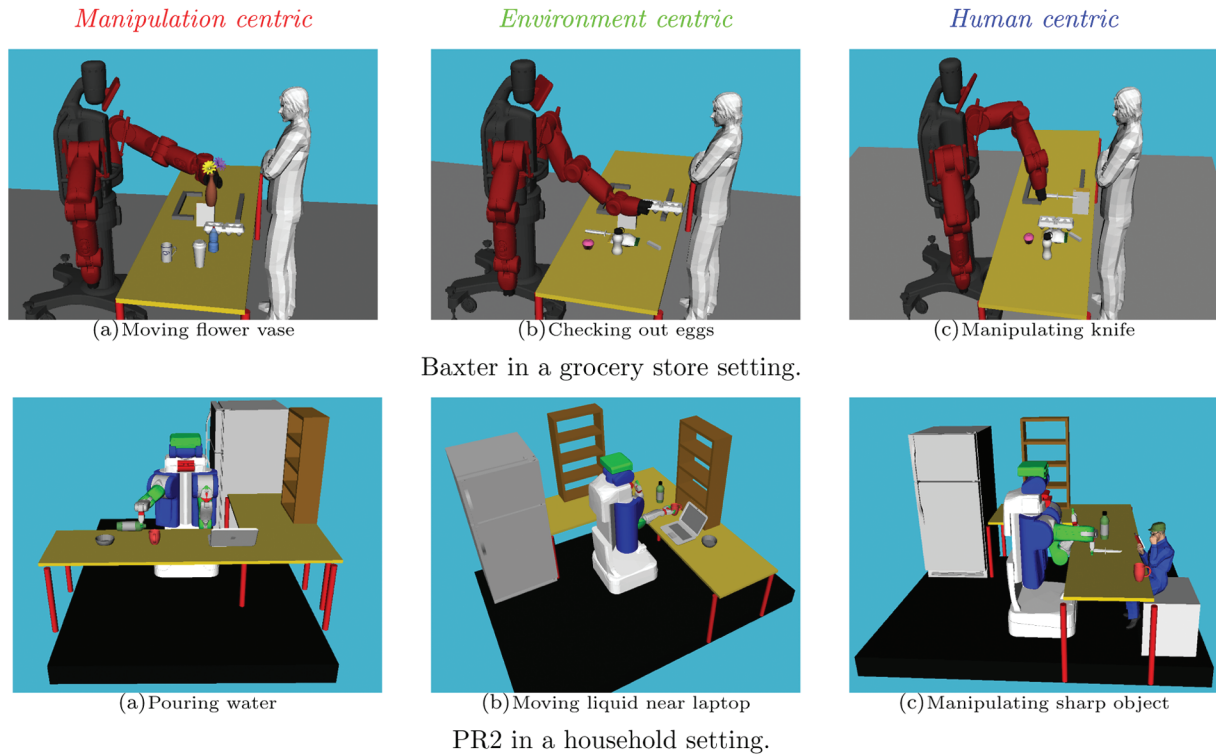
(a) Moving flower vase   (b) Checking out eggs   (c) Manipulating knife

Baxter in a grocery store setting.

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

PR2 in a household setting.

Fig. 8. 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.

most commonly occurring activities in household and grocery stores:

1. *Manipulation centric*: These activities are primarily concerned with the object being manipulated. Hence, the object's properties and the way the robot moves it in the environment are more relevant. Examples of such household activities are pouring water into a cup or inserting a pen into a pen holder, as in Figure 8(left). While in a grocery store, such activities could include moving a flower vase or moving fruits and vegetables, which could be damaged when dropped or pushed into other items. We consider *pick-and-place, pouring and inserting activities* with following objects: cup, bowl, bottle, pen, cereal box, flower vase, and tomato. Further, in every environment we place many objects, along with the object to be manipulated, to restrict simple straight line trajectories.

2. *Environment centric*: These activities are also concerned with the interactions of the object being manipulated with the surrounding objects. Our object–object interaction features (Section 5.1) allow the algorithm to learn preferences on trajectories for moving fragile objects such as egg cartons or moving liquid near electronic devices, as in Figure 8(middle). We

consider moving fragile items such as egg carton, heavy metal boxes near a glass table, water near a laptop and other electronic devices.

3. *Human centric*: Sudden movements by the robot put the human in danger of getting hurt. We consider activities where a robot manipulates sharp objects such as knife, as in Figure 8(right), moves a hot coffee cup or a bowl of water with a human in vicinity.

6.1.2. Experiment setting. Through experiments we will study the following.

- *Generalization*: Performance of robot on tasks that it has not seen before.
- *No demonstrations*: Comparison of TPP with algorithms that also learn in the absence of expert's demonstrations.
- *Feedback*: Effectiveness of different kinds of user feedback in absence of expert's demonstrations.

6.1.3. Baseline algorithms. We evaluate algorithms that learn preferences from online feedback under two settings: (a) *untrained*, where the algorithms learn preferences for a new task from scratch without observing any previous feedback; (b) *pre-trained*, where the algorithms are pre-trained

---

Algorithm 2. MMP-online.

Initialize $w_O^{(1)} \leftarrow 0$, $w_E^{(1)} \leftarrow 0$, $\mathcal{T} = \{\}$
  for $t = 1$ to T **do**
    Sample trajectories $\{y^{(1)},\dots,y^{(n)}\}$
    $y_t = argmax_y s(x_t, y; w_O^{(t)}, w_E^{(t)})$
    Obtain user feedback $\bar{y}_t$
    $\mathcal{T} = \mathcal{T} \cup \{(x_t, \bar{y}_t)\}$
    $w_O^{(t+1)}, w_E^{(t+1)} = \text{Train} - \text{SSVM}(\mathcal{T})$ (Joachims   et   al.
    (Joachims et al., 2009))
  end for

---

on other similar tasks, and then adapt to a new task. We compare the following algorithms.

- *Geometric*: The robot plans a path, independent of the task, using a bi-directional RRT (BiRRT) (LaValle and Kuffner, 2001) planner.
- *Manual*: The robot plans a path following certain manually coded preferences.
- *TPP*: Our algorithm, evaluated under both *untrained* and *pre-trained* settings.
- *MMP-online*: This is an online implementation of the Maximum Margin Planning (MMP) (Ratliff et al., 2006, 2009a) algorithm. MMP attempts to make an expert's trajectory better than any other trajectory by a margin. It can be interpreted as a special case of our algorithm with 1-informative, i.e. optimal feedback. However, directly adapting MMP (Ratliff et al., 2006) to our experiments poses two challenges: (i) we do not have knowledge of the optimal trajectory; and (ii) the state space of the manipulator we consider is too large, discretizing which makes intractable to train MMP.

To ensure a fair comparison, we follow the MMP algorithm from (Ratliff et al., 2006, 2009a) and train it under similar settings as TPP. Algorithm 2 shows our implementation of MMP-online. It is very similar to TPP (Algorithm 1) but with a different parameter update step. Since both algorithms only observe user feedback and not demonstrations, MMP-online treats each feedback as a proxy for optimal demonstration. At every iteration MMP-online trains a structural support vector machine (SSVM) (Joachims et al., 2009) using all previous feedback as training examples, and use the learned weights to predict trajectory scores in the next iteration. Since the argmax operation is performed on a set of trajectories it remains tractable. We quantify closeness of trajectories by the L_2-norm of the difference in their feature representations, and choose the regularization parameter C for training SSVM in hindsight, giving an unfair advantage to MMP-online.

6.1.4. Evaluation metrics. In addition to performing a user study (Section 6.4), we also designed two datasets to quantitatively evaluate the performance of our online algorithm.

We obtained experts labels on 1300 trajectories in a grocery setting and 2100 trajectories in a 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 evaluate performance of algorithms by measuring how well they rank trajectories, that is, trajectories with higher Likert score should be ranked higher. To quantify the quality of a ranked list of trajectories we report normalized discounted cumulative gain (nDCG) (Manning et al., 2008), a criterion popularly used in information retrieval for document ranking. In particular we report nDCG at positions 1 and 3, equation (6). 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). For a given ranked list of items (trajectories here) nDCG at position k is defined as

$$DCG@k = \sum_{i=1}^{k} \frac{l_i}{\log_2(i+1)} \qquad (5)$$

$$nDCG@k = \frac{DCG@k}{IDCG@k} \qquad (6)$$

where l_i is the Likert score of the item at position i in the ranked list. *IDCG* is the *DCG* value of the best possible ranking of items. It is obtained by ranking items in decreasing order of their Likert score.

6.2. Results and discussion

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

6.2.1. How well does TPP generalize to new tasks? To study generalization of preference feedback 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 9 shows nDCG@3 plots averaged over tasks for all types of activities for both household and grocery store settings.[5] TPP-pre-trained starts-off with higher nDCG@3 values than TPP-untrained in all three cases. However, as more feedback is provided, the performance of both algorithms improves, and they eventually give identical performance. We further observe that generalizing to tasks with both new environment and object is harder than when only one of them changes.
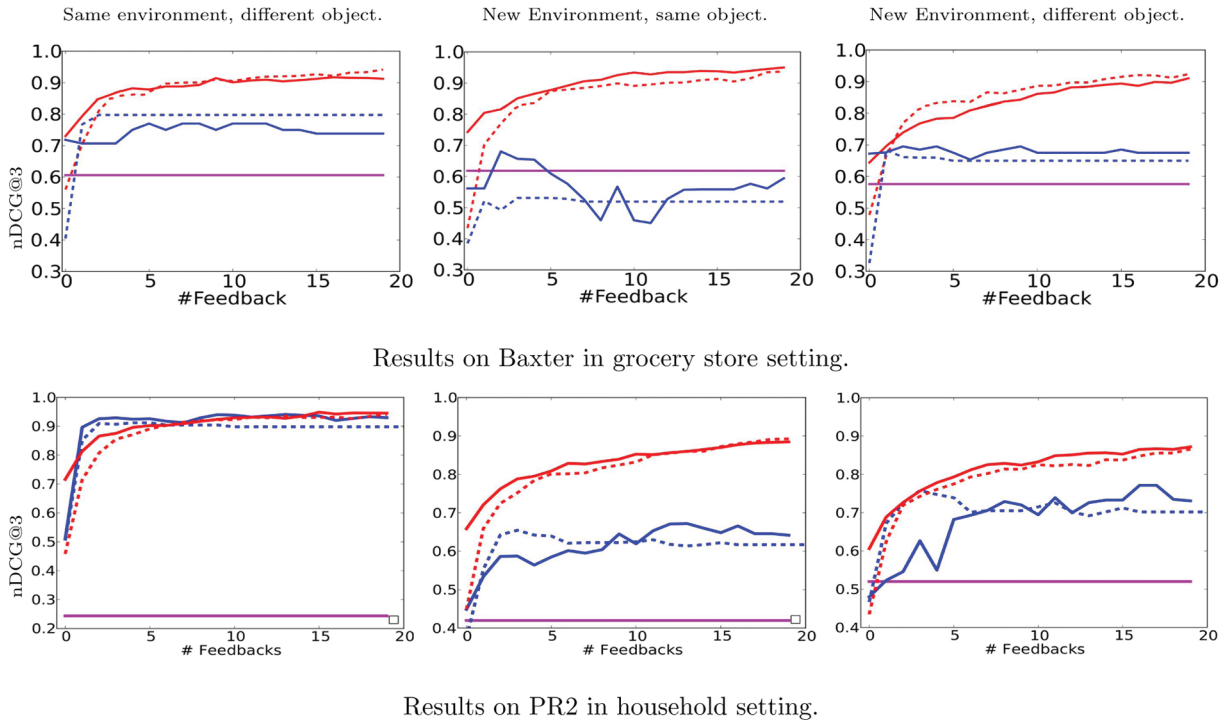
Results on Baxter in grocery store setting.

Results on PR2 in household setting.

Fig. 9. Study of generalization with change in object, environment and both. Manual (pink), pre-trained MMP-online (blue —), untrained MMP-online (blue —), pre-trained TPP (red —), untrained TPP (red —).

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

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	0.46 (0.48)	0.45 (0.39)	0.31 (0.30)	0.40 (0.39)	0.36 (0.54)	0.43 (0.38)	0.36 (0.27)	0.38 (0.40)
Manual	0.61 (0.62)	0.77 (0.77)	0.33 (0.31)	0.57 (0.57)	0.53 (0.55)	0.39 (0.53)	0.40 (0.37)	0.44 (0.48)
MMP-online	0.47 (0.50)	0.54 (0.56)	0.33 (0.30)	0.45 (0.46)	0.83 (0.82)	0.42 (0.51)	0.36 (0.33)	0.54 (0.55)
TPP	**0.88 (0.84)**	**0.90 (0.85)**	**0.90 (0.80)**	**0.89 (0.83)**	**0.93 (0.92)**	**0.85 (0.75)**	**0.78 (0.66)**	**0.85 (0.78)**

6.2.2. How does TPP compare to MMP-online? MMP-online while training assumes all user feedback is optimal, and hence over time it accumulates many contradictory/ sub-optimal training examples. We empirically observe that MMP-online generalizes better in grocery store setting than the household setting (Figure 9), however under both settings its performance remains much lower than TPP. This also highlights the sensitivity of MMP to sub-optimal demonstrations.

6.2.3. How does TPP compare to manual? For the manual baseline 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 9) that 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 iterations in untrained setting. For both household and grocery activities, TPP performs better than other baseline algorithms.

6.2.4. 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.

- *Click-one-to-replace-top*: The 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.
- *Click-one-from-five*: The top five trajectories are shown and user clicks on the one they think is the best after watching all five of them.
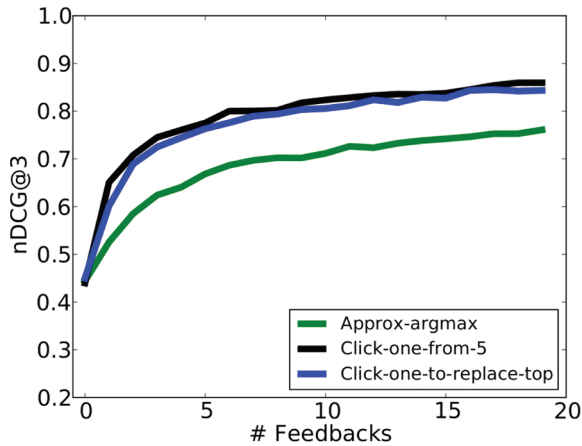- *Approximate-argmax*: This is a weaker feedback, here instead of presenting top ranked trajectories, five

Fig. 10. Study of re-rank feedback on Baxter for the grocery store setting.

Fig. 11. Comparison with fully supervised Oracle-svm on Baxter for the grocery store setting.

random trajectories are selected as candidates. The user selects the best trajectory among these five candidates. This simulates a situation when computing an argmax over trajectories is prohibitive and therefore approximated.

Figure 10 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 iterations to learn preferences. In particular, click-one-to-replace-top and click-one-from-five are more informative than approximate-argmax and therefore require less feedback to reach a given nDCG@1 value. Approximate-argmax improves slowly since it is least informative. In all three cases the feedback is α-informative, for some $\alpha > 0$, therefore TPP-untrained eventually learns the user's preferences.

6.3. Comparison with fully supervised algorithms

The algorithms discussed so far only observes ordinal feedback where the users iteratively improves upon the proposed trajectory. In this section we compare TPP with a fully supervised algorithm that observes expert's labels while training. Eliciting such expert labels on the large space of trajectories is not realizable in practice. However, empirically it nonetheless provides an upper-bound on the generalization to new tasks. We refer to this algorithm as *Oracle-svm* and it learns to rank trajectories using SVM-rank (Joachims, 2006). Since expert labels are not available while prediction, on test set Oracle-svm predicts once and does not learn from user feedback.

Figure 11 compares TPP and Oracle-svm on new tasks. Without observing any feedback on new tasks Oracle-svm performs better than TPP. However, after few feedback iterations TPP improves over Oracle-svm, which is not updated since it requires expert's labels on test set. On
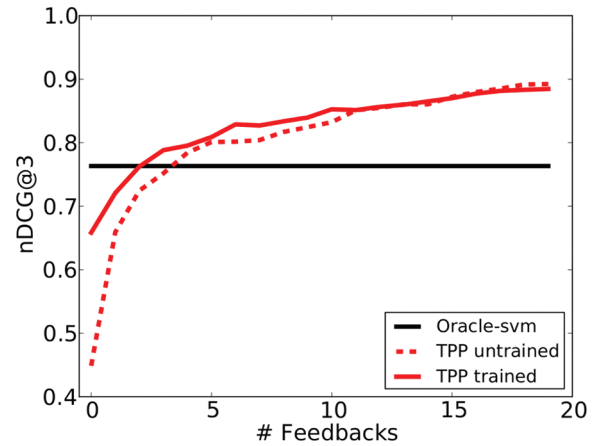
average, we observe, it takes five feedback iterations for TPP to improve over Oracle-svm. Furthermore, LfD can be seen as a special case of Oracle-svm where, instead of providing an expert label for every sampled trajectory, the expert directly demonstrates the optimal trajectory.

6.4. 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 a proof-of-concept of our approach in real world robotic scenarios, and that the combination of re-ranking and zero-G/interactive feedback allows users to train the robot in a few feedback iterations.

6.4.1. Experiment setup. In this study, users not associated with this work used our system to train PR2 and Baxter on household and grocery checkout tasks, respectively. Five users independently trained Baxter, by providing zero-G feedback kinesthetically on the robot, and re-rank feedback in a simulator. Two users participated in the study on PR2. On PR2, in place of zero-G, users provided interactive way-point correction feedback in the Rviz simulator. The users were undergraduate students. Further, both users training PR2 on household tasks were familiar with Rviz-ROS.[6] 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. We also recorded the total time a user spent on each task, from the start of training until the user was satisfied with the top ranked trajectory. This includes time taken for both re-rank and zero-G feedback.

Table 2. Learning statistics for each user. Self- and cross-scores of the final learned trajectories. The numbers inside brackets are standard deviation. (Top) Results for grocery store on Baxter. (Bottom) Household setting on PR2.

User	# Re-ranking feedback	# Zero-G feedback	Average time (min)	Trajectory-quality	
				self	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 feedback	# Interactive feedbacks	Trajectory-quality	
			self	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)

6.4.2. Is re-rank feedback easier to elicit from users than zero-G or interactive? In our user study, on average a user took three re-rank and two zero-G feedback per task to train a robot (Table 2). From this we conjecture, that 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 count of zero-G (interactive) feedback with task difficulty suggests (Figure 12(right)), users rely more on zero-G feedback for difficult tasks since it allows precisely rectifying erroneous way-points. Figures 13 and 14 show two example trajectories learned by a user.

6.4.3. How many feedback iterations a user takes to improve over Oracle-svm? Figure 12(left) shows that the quality of trajectory improves with feedback. On average, a user took five feedback to improve over Oracle-svm, which is also consistent with our quantitative analysis (Section 6.3). In grocery setting, users 4 and 5 were critical towards trajectories learned by Oracle-svm and gave them low scores. This indicates a possible mismatch in preferences between our expert (on whose labels Oracle-svm was trained) and users 4 and 5.

6.4.4. How do users' unobserved score functions vary? An average difference of 0.6 between users' self- and cross-score (Table 2) in the 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, future work might explore coactive learning for satisfying user population, similar to Raman and Joachims (2013). For household setting, the sample size is too small to draw a similar conclusion.

6.4.5. How long does it take for users to train a robot? We report training time for only the grocery store setting, because the interactive feedback in the household setting requires users with experience in Rviz-ROS. Further, we observed that users found it difficult to modify the robot's joint angles in a simulator to their desired configuration. In the grocery checkout setting, among all of 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 interfaces (Shneiderman and Plaisant, 2010) could further reduce this time. For example, simultaneous visualization of top ranked trajectories instead of sequentially showing them to users (the scheme we currently adopt) could bring down the time for re-rank feedback. Despite its limited size, through our user study we show that 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 and code, visit `http://pr.cs.cornell.edu/coactive/`

7. Conclusion and future work

When manipulating objects in human environments, it is important for robots to plan motions that follow users' 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 teaching robots these preferences through iterative improvements from non-expert users. Unlike in standard LfD 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 (with Baxter) settings. 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.
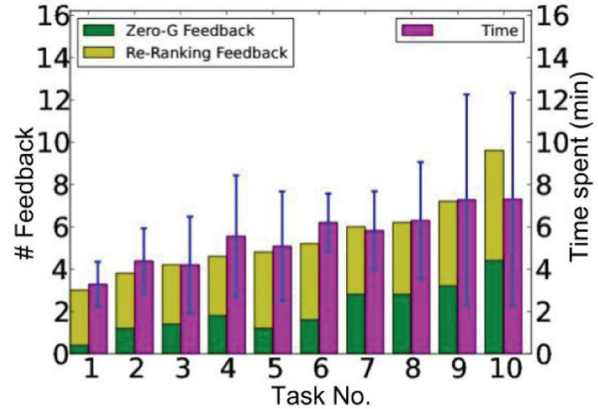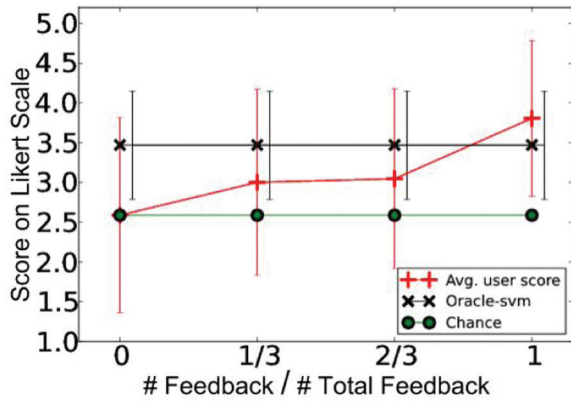
Grocery store setting on Baxter.
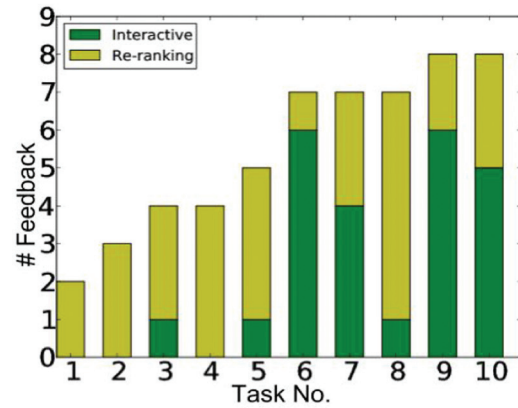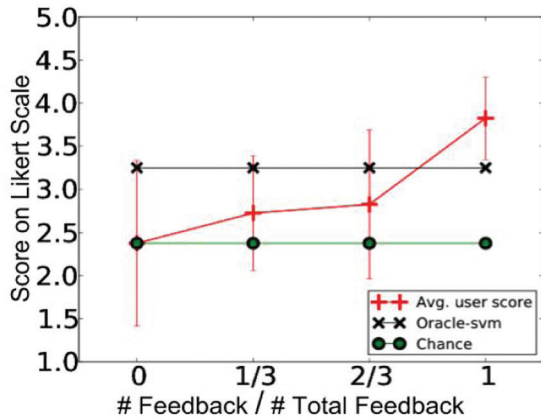
Household setting on PR2.

Fig. 12. (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.
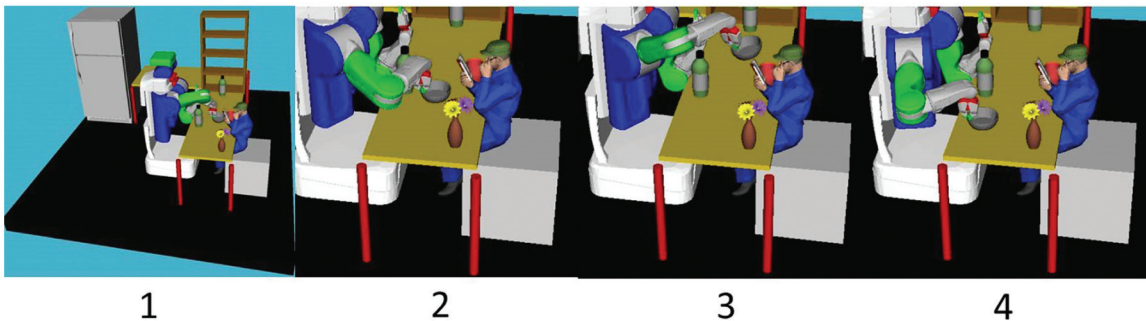

Fig. 13. Trajectories for moving a bowl of water in the presence of a human. Without learning robot plans an undesirable trajectory and moves bowl over the human (waypoints 1–3–4). After six user feedback the robot learns the desirable trajectory (waypoints 1–2–4).

Future research could extend coactive learning to situations with uncertainty in object pose and attributes. Under uncertainty the TPP will admit a belief space update form, and theoretical guarantees will also be different. Coactive feedback might also find use in other interesting robotic applications such as assistive cars, where a car learns from
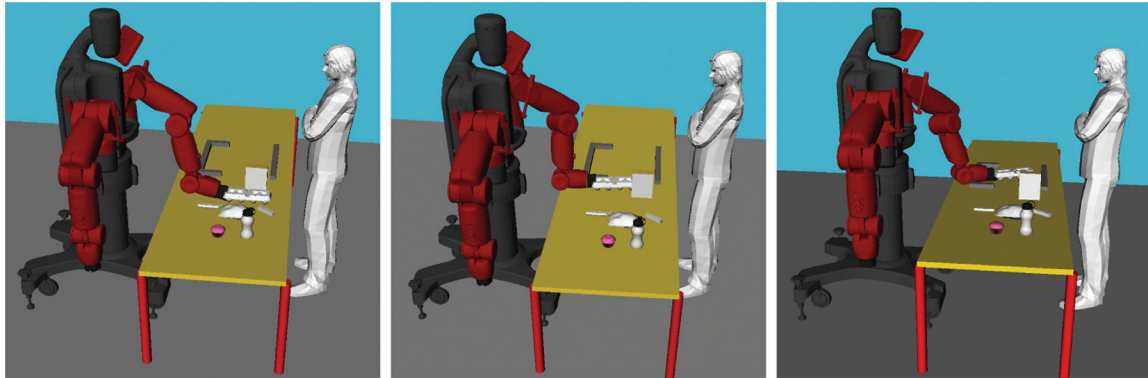
Fig. 14. The learned trajectory for moving an egg carton. Since eggs are fragile the robot moves the carton near the table surface. (Left) Start of the trajectory. (Middle) Intermediate waypoint with egg close to the table surface. (Right) End of the trajectory.

humans steering actions. Scaling up coactive feedback by crowd-sourcing and exploring other forms of easy-to-elicit learning signals are also potential future directions.

Notes

1. 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.
2. In this work, our goal is to relax the assumption of unbiased and close to optimal feedback. We therefore assume complete knowledge of the environment for our algorithm, and for the algorithms we compare against. In practice, such knowledge can be extracted using an object attribute labeling algorithms such as in Koppula et al. (2011).
3. We query PQP collision checker plugin of OpenRave for these distances.
4. 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 such as CHOMP (Ratliff et al., 2009b) or optimal planners such as RRT* (Karaman and Frazzoli, 2010) to produce reasonably good trajectories.
5. Similar results were obtained with the nDCG@1 metric.
6. 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 a grocery store setting.

References

Abbeel P, Coates A and Ng AY (2010) Autonomous helicopter aerobatics through apprenticeship learning. *The International Journal of Robotics Research* 29(13): 1608–1639.

Akgun B, Cakmak M, Jiang K and Thomaz AL (2012) Keyframe-based learning from demonstration. *International Journal of Social Robotics* 4(4): 343–355.

Alterovitz R, Siméon T and Goldberg K (2007) The stochastic motion roadmap: a sampling framework for planning with Markov motion uncertainty. In: *Proceedings of robotics: science and systems*.

Argall BD, Chernova S, Veloso M and Browning B (2009) A survey of robot learning from demonstration. *Robotics and Autonomous Systems* 57(5): 469–483.

Berenson D, Abbeel P and Goldberg K (2012) A robot path planning framework that learns from experience. In: *Proceedings of the international conference on robotics and automation*.

Berg JVD, Abbeel P and Goldberg K (2010) LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information. In: *Proceedings of robotics: science and systems*.

Bhattacharya S, Likhachev M and Kumar V (2011) Identification and representation of homotopy classes of trajectories for search-based path planning in 3D. In: *Proceedings of robotics: science and systems*.

Bischoff R, Kazi A and Seyfarth M (2002) The morpha style guide for icon-based programming. In: *Proceedings 11th IEEE international workshop on RHIC*.

Calinon S, Guenter F and Billard A (2007) On learning, representing, and generalizing a task in a humanoid robot. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 37(2): 286–298.

Cohen BJ, Chitta S and Likhachev M (2010) Search-based planning for manipulation with motion primitives. In: *Proceedings of the international conference on robotics and automation*, pp. 2902–2908.

D Ferguson D and Stentz A (2006) Anytime RRTs. In: *Proceedings of the IEEE/RSJ conference on intelligent robots and systems*.

Dey D, Liu TY, Hebert M and Bagnell JA (2012) Contextual sequence prediction with application to control library optimization. In: *Proceedings of robotics: science and systems*.

Diankov R (2010) *Automated Construction of Robotic Manipulation Programs*. PhD Thesis, CMU, RI.

Dragan A, Lee K and Srinivasa S (2013) Legibility and predictability of robot motion. In: *HRI'13 Proceedings of the 8th ACM/IEEE international conference on human–robot interaction.*

Dragan A and Srinivasa S (2013) Generating legible motion. In: *Proceedings of robotics: science and systems.*

Erickson LH and LaValle SM (2009) Survivability: Measuring and ensuring path diversity. In: *Proceedings of the international conference on robotics and automation.*

Ettlin A and Bleuler H (2006) Randomised rough-terrain robot motion planning. In: *Proceedings of the IEEE/RSJ conference on intelligent robots and systems.*

Gossow D, Hershberger ALD and Ciocarlie M (2011) Interactive markers: 3-D user interfaces for ROS applications [ROS topics]. *IEEE Robotics and Automation Magazine* 18(4): 14–15.

Green CJ and Kelly A (2011) Toward optimal sampling in the space of paths. In: *Robotics Research (Springer Tracts in Advanced Robotics*, Vol. 66). New York: Springer, pp. 281–292.

Jaillet L, Cortés J and Siméon T (2010) Sampling-based path planning on configuration-space costmaps. *IEEE Transactions on Robotics* 26(4): 635–646.

Jain A, Sharma S and Saxena A (2013a) Beyond geometric path planning: Learning context-driven user preferences via suboptimal feedback. In: *Proceedings of the international symposium on robotics research.*

Jain A, Wojcik B, Joachims T and Saxena A (2013b) Learning trajectory preferences for manipulators via iterative improvement. In: *Advances in Neural Information Processing Systems.*

Jiang Y, Lim M, Zheng C and Saxena A (2012) Learning to place new objects in a scene. *The International Journal of Robotics Research* 31(9): 1021–1043.

Joachims T (2006) Training linear svms in linear time. In: *Proceedings of the ACM special interest group on knowledge discovery and data mining.*

Joachims T, Finley T and Yu CNJ (2009) Cutting-plane training of structural SVMs. *Machine Learning* 77(1): 27–59.

Karaman S and Frazzoli E (2010) Incremental sampling-based algorithms for optimal motion planning. In: *Proceedings of robotics: science and systems.*

Karaman S and Frazzoli E (2011) Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research* 30(7): 846–894.

Kitani KM, Ziebart BD, Bagnell JA and Hebert M (2012) Activity forecasting. In: *Proceedings of the European conference on computer vision.*

Klingbeil E, Rao D, Carpenter B, Ganapathi V, Ng AY and Khatib O (2011) Grasping with application to an autonomous checkout robot. In: *Proceedings of the international conference on robotics and automation.*

Kober J and Peters J (2011) Policy search for motor primitives in robotics. *Machine Learning* 84(1–2): 171–203.

Koppula H, Anand A, Joachims T and Saxena A (2011) Semantic labeling of 3D point clouds for indoor scenes. In: *Advances in Neural Information Processing Systems.*

Kretzschmar H, Kuderer M and Burgard W. Learning to predict trajectories of cooperatively navigating agents. In *Proceedings of the International Conference on Robotics and Automation*2014.

LaValle SM and Kuffner JJ (2001) Randomized kinodynamic planning. *The International Journal of Robotics Research* 20(5): 378–400.

Lenz I, Lee H and Saxena A (2013) Deep learning for detecting robotic grasps. In: *Proceedings of robotics: science and systems.*

Leven P and Hutchinson S (2003) Using manipulability to bias sampling during the construction of probabilistic roadmaps. *IEEE Transactions on Robotics and Automation* 19(6): 1020–1026.

Levine S and Koltun V (2012) Continuous inverse optimal control with locally optimal examples. In: *Proceedings of the international conference on machine learning.*

Mainprice J, Sisbot EA, Jaillet L, Cortés J, Alami R and Siméon T (2011) Planning human-aware motions using a sampling-based costmap planner. In: *Proceedings of the international conference on robotics and automation.*

Manning CD, Raghavan P and Schütze H (2008) *Introduction to information retrieval,* volume 1. Cambridge: Cambridge University Press.

Nikolaidis S and Shah J (2012) Human-robot teaming using shared mental models. In: *HRI, Workshop on human–agent–robot teamwork.*

Nikolaidis S and Shah J (2013) Human-robot cross-training: Computational formulation, modeling and evaluation of a human team training strategy. In: *IEEE/ACM ICHRI.*

Phillips M, Cohen B, Chitta S and Likhachev M (2012) E-graphs: Bootstrapping planning with experience graphs. In: *Proceedings of robotics: science and systems.*

Raman K and Joachims T (2013) Learning socially optimal information systems from egoistic users. In: *Proceedings of the European conference on machine learning.*

Ratliff N (2009) *Learning to Search: Structured Prediction Techniques for Imitation Learning.* PhD Thesis, CMU, RI.

Ratliff N, Bagnell JA and Zinkevich M (2006) Maximum margin planning. In: *Proceedings of the international conference on machine learning.*

Ratliff N, Bagnell JA and Zinkevich M (2007) (online) subgradient methods for structured prediction. In: *Proceedings of the international conference on artificial intelligence and statistics.*

Ratliff N, Silver D and Bagnell JA (2009a) Learning to search: Functional gradient techniques for imitation learning. *Autonomous Robots* 27(1): 25–53.

Ratliff N, Zucker M, Bagnell JA and Srinivasa S (2009b) CHOMP: Gradient optimization techniques for efficient motion planning. In: *Proceedings of the international conference on robotics and automation.*

Ross S, Zhou J, Yue Y, Dey D and Bagnell JA (2013) Learning policies for contextual submodular prediction. *Proceedings of the international conference on machine learning.*

Saxena A, Driemeyer J and Ng AY (2008) Robotic grasping of novel objects using vision. *The International Journal of Robotics Research* 27(2): 157–173.

Schulman J, Ho J, Lee A, Awwal I, Bradlow H and Abbeel P (2013) Finding locally optimal, collision-free trajectories with sequential convex optimization. In: *Proceedings of robotics: science and systems.*

Shivaswamy P and Joachims T (2012) Online structured prediction via coactive learning. In: *Proceedings of the international conference on machine learning.*

Shneiderman B and Plaisant C (2010) *Designing The User Interface: Strategies for Effective Human–Computer Interaction.* Reading, MA: Addison-Wesley Publication.

Silver D, Bagnell JA and Stentz A (2010) Learning from demonstration for autonomous navigation in complex unstructured terrain. *The International Journal of Robotics Research* 29(12): 1565–1592.

Sisbot EA, Marin LF and Alami R (2007a) Spatial reasoning for human robot interaction. In: *Proceedings of the IEEE/RSJ conference on intelligent robots and systems*.

Sisbot EA, Marin-Urias LF, Alami R and Simeon T (2007b) A human aware mobile robot motion planner. *IEEE Transactions on Robotics* 23(5): 874–883.

Stopp A, Horstmann S, Kristensen S and Lohnert F (2001) Towards interactive learning for manufacturing assistants. In: *Proceedings 10th IEEE international workshop on RHIC*.

Sucan IA, Moll M and Kavraki LE (2012) The Open Motion Planning Library. *IEEE Robotics and Automation Magazine* 19(4): 72–82.

Tamane K, Revfi M and Asfour T (2013) Synthesizing object receiving motions of humanoid robots with human motion database. In: *Proceedings of the international conference on robotics and automation*.

Vernaza P and Bagnell JA (2012) Efficient high dimensional maximum entropy modeling via symmetric partition functions. In: *Advances in Neural Information Processing Systems*.

Wilson A, Fern A and Tadepalli P (2012) A Bayesian approach for policy learning from trajectory preference queries. In: *Advances in Neural Information Processing Systems*.

Zacharias F, Schlette C, Schmidt F, Borst C, Rossmann J and Hirzinger G (2011) Making planned paths look more human-like in humanoid robot manipulation planning. In: *Proceedings of the international conference on robotics and automation*.

Ziebart BD, Maas A, Bagnell JA and Dey AK (2008) Maximum entropy inverse reinforcement learning. In: *AAAI*.

Zucker M, Ratliff N, Dragan AD, et al. (2013) CHOMP: Covariant Hamiltonian optimization for motion planning. *The International Journal of Robotics Research* 32(9–10): 1164–1193.

Appendix A: Proof for average regret

This proof builds upon Shivaswamy and Joachims (2012).

We assume the user hidden score function $s^*(x, y)$ is contained in the family of scoring functions $s(x, y; w_O^*, w_E^*)$ for some unknown w_O^* and w_E^*. Average regret for TPP over T rounds of interactions can be written as

$$REG_T = \frac{1}{T} \sum_{t=1}^{T} (s^*(x_t, y_t^*) - s^*(x_t, y_t))$$
$$= \frac{1}{T} \sum_{t=1}^{T} (s(x_t, y_t^*; w_O^*, w_E^*) - s(x_t, y_t; w_O^*, w_E^*))$$

We further assume the feedback provided by the user is strictly α-informative and satisfy following inequality:

$$s(x_t, \bar{y}_t; w_O^*, w_E^*) \geq s(x_t, y_t; w_O^*, w_E^*) + \alpha[s(x_t, y_t^*; w_O^*, w_E^*) - s(x_t, y_t; w_O^*, w_E^*)] - \xi_t \quad (7)$$

Later we relax this constraint and requires it to hold only in expectation.

This definition states that the user feedback should have a score of $\bar{y}_t$ that is higher than that of y_t by a fraction α $\in (0, 1]$ of the maximum possible range $s(x_t, y_t^*; w_O^*, w_E^*) - s(x_t, y_t; w_O^*, w_E^*)$.

Theorem 1. *The average regret of trajectory preference perceptron receiving strictly α-informative feedback can be upper bounded for any $[w_O^*; w_E^*]$ as follows:*

$$REG_T \leq \frac{2C\|[w_O^*; w_E^*]\|}{\alpha\sqrt{T}} + \frac{1}{\alpha T} \sum_{t=1}^{T} \xi_t \quad (8)$$

where C is constant such that $\|[\phi_O(x, y); \phi_E(x, y)]\|_2 \leq C$.

Proof. After T rounds of feedback, using weight update equations of w_E and w_O we can write

$$w_O^* \cdot w_O^{(T+1)} = w_O^* \cdot w_O^{(T)} + w_O^* \cdot (\phi_O(x_T, \bar{y}_T) - \phi_O(x_T, y_T))$$
$$w_E^* \cdot w_E^{(T+1)} = w_E^* \cdot w_E^{(T)} + w_E^* \cdot (\phi_E(x_T, \bar{y}_T) - \phi_E(x_T, y_T))$$

Adding the two equations and recursively reducing the right gives:

$$w_O^* \cdot w_O^{(T+1)} + w_E^* \cdot w_E^{(T+1)} = \sum_{t=1}^{T} (s(x_t, \bar{y}_t; w_O^*, w_E^*) - s(x_t, y_t; w_O^*, w_E^*)) \quad (9)$$

Using Cauchy–Schwarz inequality the left-hand side of equation (9) can be bounded as

$$w_O^* \cdot w_O^{(T+1)} + w_E^* \cdot w_E^{(T+1)} \leq \|[w_O^*; w_E^*]\| \|[w_O^{(T+1)}; w_E^{(T+1)}]\| \quad (10)$$

$\|[w_O^{(T+1)}; w_E^{(T+1)}]\|$ can be bounded by using weight update equations:

$$w_O^{(T+1)} \cdot w_O^{(T+1)} + w_E^{(T+1)} \cdot w_E^{(T+1)} = w_O^{(T)} \cdot w_O^{(T)} + w_E^{(T)} \cdot w_E^{(T)}$$
$$+ 2w_O^{(T)} \cdot (\phi_O(x_T, \bar{y}_T) - \phi_O(x_T, y_T))$$
$$+ 2w_E^{(T)} \cdot (\phi_E(x_T, \bar{y}_T) - \phi_E(x_T, y_T))$$
$$+ (\phi_O(x_T, \bar{y}_T) - \phi_O(x_T, y_T)) \cdot (\phi_O(x_T, \bar{y}_T) - \phi_O(x_T, y_T))$$
$$+ (\phi_E(x_T, \bar{y}_T) - \phi_E(x_T, y_T)) \cdot (\phi_E(x_T, \bar{y}_T) - \phi_E(x_T, y_T))$$
$$\leq w_O^{(T)} \cdot w_O^{(T)} + w_E^{(T)} \cdot w_E^{(T)} + 4C^2 \leq 4C^2 T \quad (11)$$

$$\therefore \|[w_O^{(T+1)}; w_E^{(T+1)}]\| \leq 2C\sqrt{T} \quad (12)$$

Equation (11) follows from the fact that $s(x_T, y_T; w_O^{(T)}, w_E^{(T)}) > s(x_T, \bar{y}_T; w_O^{(T)}, w_E^{(T)})$ and $\|[\phi_O(x, y); \phi_E(x, y)]\|_2 \leq C$. Using equations (10) and (12) gives following bound on (9):

$$\sum_{t=1}^{T} (s(x_t, \bar{y}_t; w_O^*, w_E^*) - s(x_t, y_t; w_O^*, w_E^*)) \quad (13)$$
$$\leq 2C\sqrt{T}\|[w_O^*; w_E^*]\|$$

Assuming strictly α-informative feedback and re-writing equation (7) as

$$s(x_t, y_t^*; w_O^*, w_E^*) - s(x_t, y_t; w_O^*, w_E^*)$$
$$\leq \frac{1}{\alpha}((s(x_t, \bar{y}_t; w_O^*, w_E^*) - s(x_t, y_t; w_O^*, w_E^*)) - \xi_t) \qquad (14)$$

Combining equations (13) and (14) gives the bound on average regret (8).

Appendix B: Proof for expected regret

We now show the regret bounds for TPP under a weaker feedback assumption, expected α-informative feedback:

$$E_t[s(x_t, \bar{y}_t; w_O^*, w_E^*)] \geq s(x_t, y_t; w_O^*, w_E^*)$$
$$+ \alpha[s(x_t, y_t^*; w_O^*, w_E^*) - s(x_t, y_t; w_O^*, w_E^*)] - \xi_t$$

where the expectation is under choices $\bar{y}_t$ when y_t and x_t are known.

Corollary 2. *The expected regret of TPP receiving expected α-informative feedback can be upper bounded for any $[w_O^*; w_E^*]$ as follows:*

$$E[REG_T] \leq \frac{2C\left\|[w_G^*; w_O^*]\right\|}{\alpha\sqrt{T}} + \frac{1}{\alpha T}\sum_{t=1}^{T} \bar{\xi}_t \qquad (15)$$

Proof: Taking expectation on both sides of equations (9), (10) and (11) yields following equations respectively:

$$E[w_O^* \cdot w_O^{(T+1)} + w_E^* \cdot w_E^{(T+1)}] \quad = \sum_{t=1}^{T} E[(s(x_t, \bar{y}_t; w_O^*, w_E^*) - s(x_t, y_t; w_O^*, w_E^*))] \qquad (16)$$

$$E[w_O^* \cdot w_O^{(T+1)} + w_E^* \cdot w_E^{(T+1)}]$$
$$\leq \left\|[w_O^*; w_E^*]\right\| E\left[\left\|[w_O^{(T+1)}; w_E^{(T+1)}]\right\|\right]$$

$$E[w_O^{(T+1)} \cdot w_O^{(T+1)} + w_E^{(T+1)} \cdot w_E^{(T+1)}] \leq 4C^2 T$$

Applying Jensen's inequality on the concave function $\sqrt{\cdot}$ we obtain

$$E[w_O^* \cdot w_O^{(T+1)} + w_E^* \cdot w_E^{(T+1)}]$$
$$\leq \left\|[w_O^*; w_E^*]\right\| E\left[\left\|[w_O^{(T+1)}; w_E^{(T+1)}]\right\|\right]$$
$$\leq \left\|[w_O^*; w_E^*]\right\| \sqrt{E[w_O^{(T+1)} \cdot w_O^{(T+1)} + w_E^{(T+1)} \cdot w_E^{(T+1)}]}$$

Using (16) gives the following bound:

$$\sum_{t=1}^{T} E[s(x_t, \bar{y}_t; w_O^*, w_E^*) - s(x_t, y_t; w_O^*, w_E^*)]$$
$$\leq 2C\sqrt{T}\left\|[w_O^*; w_E^*]\right\|$$

Now using the fact that the user feedback is expected α-informative gives the regret bound (15).