Layered Hybrid Inverse Optimal Control for Learning Robot Manipulation from Demonstration

Arunkumar Byravan[1], Mathew Montfort[2], Brian Ziebart[2], Byron Boots[3] and Dieter Fox[1*]

Abstract

Inverse optimal control (IOC) is a powerful approach for learning robotic controllers from demonstration that estimates a cost function which rationalizes demonstrated control trajectories. Unfortunately, its applicability is difficult in settings where optimal control can only be solved approximately. Local IOC approaches rationalize demonstrated trajectories based on a linear-quadratic approximation around a good reference trajectory (i.e., the demonstrated trajectory itself). Without this same reference trajectory, however, dissimilar control results. We address the complementary problem of using IOC to find appropriate reference trajectories in these computationally challenging control tasks. After discussing the inherent difficulties of learning within this setting, we present a projection technique from the original trajectory space to a discrete and tractable trajectory space and perform IOC on this space. Control trajectories are projected back to the original space and locally optimized. We demonstrate the effectiveness of the approach with experiments conducted on a 7-degree of freedom robotic arm.

Introduction

Robotic manipulation in environments shared with humans is a difficult task. Not only must the robot succeed at grasping and placing objects, but its motion trajectories should also be predictable and legible to people who share the same workspace [1, 2]. Recent advances in planning [3] are very successful for the former objective, but less so for the latter. One reason is that even though recognizing *human-acceptable* trajectories is fairly easy, specifying an evaluation criteria that produces them is not. Fortunately, *human-acceptable* trajectories are often easy to produce through teleoperation. This makes inverse optimal control (IOC) [4, 5, 6, 7, 8] an attractive option since appropriate manipulation trajectories may be *learned* from these human demonstrations. IOC estimates a cost function for a decision process that (partially) rationalizes demonstrated sequential behavior by making it (near) optimal. The key advantage of IOC—as opposed to directly estimating a control policy—is that this estimated cost function can generalize to new situations (e.g., re-arrangement of obstacles) that are characterized by different decision processes. IOC has been employed to construct controllers and planners for helicopter flight [9], field robot navigation [7], locomotion through crowds [10], and behavior prediction tasks for human-robot interaction applications [1].

The number of degrees of freedom (DoF) for typical manipulation tasks poses significant challenges for IOC. IOC relies on solving the optimal control problem which is difficult for high-DoF problems and is therefore often solved without optimality guarantees in practice. Previous IOC applications either employ discrete, but low-dimensional, decision process representations [4, 5] or make linear-quadratic assumptions about the state dynamics and cost function [8, 11]. Unfortunately, these assumptions rarely hold in manipulation tasks. The desirable space of manipulation trajectories is often multi-modal and non-linear due to collision avoidance with discrete obstacles. Thus, the linear-quadratic assumption is limited to obtaining cost function estimates that locally rationalize demonstrated trajectories around one of these modes (in practice: the mode defined by the

[*1]Department of Computer Science & Engineering, University of Washington, [2]Department of Computer Science, University of Illinois at Chicago, [3]School of Interactive Computing, Georgia Institute of Technology

demonstrated trajectory itself) [8]. However, when the controller applies the resulting cost function to a new situation, the correct mode needed as a starting point is unknown. Other modes may produce inherently non-human-acceptable manipulation trajectories—even when the best of those is locally selected.

In this work, we propose a complementary approach to linearization and quadratic approximation for learning *human-acceptable* manipulation trajectories. First we project continuous control trajectories into a discrete graph-based representation. We then employ maximum entropy inverse optimal control [5] in the graph to find a distribution over *approximate reference modes*. Finally, we locally optimize samples from this distribution in the original trajectory space. We demonstrate the effectiveness of this approach with experiments conducted on a 7-DOF robotic arm.

Background and Related Work

Maximum entropy inverse optimal control

Maximum entropy inverse optimal control (MaxEnt IOC) [5] combines reward-based guarantees of inverse reinforcement learning [4] with predictive guarantees of robust statistical estimation [12, 13]. It obtains the stochastic policy that is least biased while still matching feature counts [4]. Its resulting predictive policy estimate, $P(a_t|s_t) \propto e^{Q(s_t,a_t)}$, is recursively defined using a softened version of the Bellman equation:

$$Q(s_t, a_t) \triangleq \mathbb{E}_{P(s_{t+1}|a_t,s_t)}[\underset{a_{t+1}}{\text{softmax}}\, Q(S_{t+1}, a_{t+1})] - \text{cost}_{\theta,\mathbf{f}}(s_t, a_t); \quad \text{cost}_{\theta,\mathbf{f}}(s_t, a_t) \triangleq \theta^{\mathrm{T}}\mathbf{f}(a_t, s_t), \quad (1)$$

where $\text{softmax}_x\, f(x) = \log \sum_x e^{f(x)}$ and $\mathbf{f}(a_t, s_t)$ is a vector of features characterizing the state-action pair. The term $\theta^{\mathrm{T}}\mathbf{f}(a_t, s_t)$ is analogous to the cost function of optimal control. Parameters θ are estimated by maximizing the (regularized) training data likelihood under the MaxEnt distribution, which for deterministic planning settings are Boltzmann distributions over state sequences:

$$P(\mathbf{s}_{1:T}, \mathbf{a}_{1:T}) = \frac{e^{-\theta^{\mathrm{T}} \sum_{t=1}^{T} \mathbf{f}(s_t,a_t)}}{\sum_{\mathbf{s}'_{1:T},\mathbf{a}'_{1:T}} e^{-\theta^{\mathrm{T}} \sum_{t=1}^{T} \mathbf{f}(s'_t,a'_t)}} \propto e^{-\text{cost}_\theta(\mathbf{s}_{1:T},\mathbf{a}_{1:T})}, \quad (2)$$

MaxEnt IOC has been developed for both discrete and continuous state and action settings. We detail both settings here.

Discrete: Discrete state-action representations can incorporate arbitrary dynamics and features, but are limited by the $\mathcal{O}(|S||A|T)$ complexity of dynamic programming algorithms that compute Eqn. 1. This limits the approach to low-dimensional settings.

Continuous: This recursion (Eqn. 1) can be analytically solved for continuous state-action representations of high-dimensional settings, but only when the dynamics are linear, $\vec{s}_{t+1} = \mathbf{A}\vec{s}_t + \mathbf{B}\vec{a}_t$, and the features are quadratic, with parameter matrix Θ, $\text{cost}(\vec{s}_{1:T}) = \mathbf{\Theta} \cdot \sum_{t=1}^{T} \vec{s}_t\vec{s}_t^{\mathrm{T}} = \sum_{t=1}^{T} \vec{s}_t^{\mathrm{T}}\mathbf{\Theta}\vec{s}_t$ [14, 11, 8]. Any continuous state-action space can be locally approximated by a linear-quadratic model around a reference trajectory [8]. However, finding an appropriate reference trajectory is an open problem.

Trajectory planning

Motion planning based on trajectory optimization has recently been very successful in solving hard, high dimensional planning problems. Algorithms like CHOMP [3] and STOMP [15] generate collision-free paths by minimizing a cost function based on trajectory smoothness and distance, while TrajOpt [16] uses sequential convex optimization with collision avoidance constraints. These methods do have drawbacks. They compute locally optimal paths with no bounds on the sub-optimality of the resulting path. Further, they do not produce trajectories that are predictive and legible by humans sharing the environment.

By *learning* the cost functions used by these algorithms, one could generate human-like trajectories that generalize sufficiently well across tasks. While the sub-optimality of the planner can be incorporated into IOC with some additional slack in the feature-matching leading to regularized learning [17]), it is the non-convexity and discontinuity of the solution's cost as a function of cost parameters that poses significant theoretical challenges for IOC. Standard gradient-based optimization methods cannot be safely employed to optimize the cost parameters of such functions since local optima created by these discontinuities leads to inappropriate parameter estimates in both theory and practice.

Two-Level Hybrid Inverse Optimal Control Approach

Motivated by the respective strengths of discrete IOC (arbitrary dynamics and features) and continuous IOC (high-dimensionality), we employ a two-level hybrid inverse optimal control approach for learning human-like robotic manipulation from demonstration. As even solving optimal control problems is computationally impractical for high-dimensional manipulation tasks, our aim is to construct a probability distribution from which sampled trajectories will be qualitatively similar to human-like demonstrations. This is in contrast to typical inverse optimal control techniques [6, 4] that attempt to make demonstrated manipulation trajectories optimal. Our approach simplifies the manipulation task into two computationally tractable models: an IOC model for a sparse discrete space that learns to avoid obstacles and follow general topological trajectory preferences; and a linear-quadratic continuous space IOC model that learns smoothness properties of human-like manipulation trajectories. We condition the trajectory distribution of our continuous model on a path through the discrete space. Figure 1 gives an overview of the steps in our approach.

The simplifications of our abstract representations intentionally ignore important aspects of the manipulation task that are computationally difficult to fully incorporate; for instance, not employing strict collision detection with certain obstacles. However, by appropriately integrating uncertainty into our trajectory distribution estimate, we can take many samples from our trajectory distribution and discard trajectories that violate any previously ignored aspects of the manipulation task. Key then is finding a representation of the manipulation task that balances computational tractability against the realizable similarity between demonstrated trajectories and the estimated trajectory distribution. We now present details on the two major components of our approach: the discrete IOC model and the continuous LQR model.

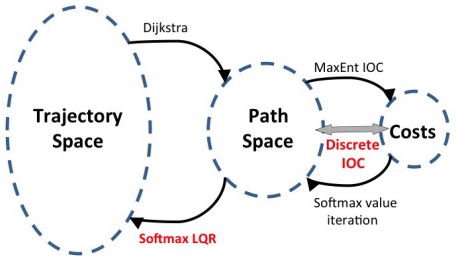

Figure 1: Two-Level Hybrid Inverse Optimal Control

Discrete IOC in path space

We construct a sparse discrete *path* space to tractably approximate our continuous *trajectory* space. This *path* space is represented as a graph in the robot's configuration space. We project the demonstrations onto this graph and use discrete MaxEnt IOC to learn a distribution over graph paths that matches the projections. Samples from this distribution are inputs to the LQR system.

Graph generation: To ensure that the graph construction is unbiased by the demonstrated trajectories, we use only the start and goal configurations to generate the graph (available to the system at test time). We discretize the straight line trajectory from start to goal and sample nodes from gaussians centered around these waypoints, accounting for joint limits. The co-variances (diagonal) and the number of sampled points from each gaussian is dependent on the distance of the gaussian center from the center of the straight line trajectory. Gaussians near the center have high variance and more points are sampled from them and those near the start and goal configurations have lower variance and give out lesser samples. Finally, we connect each node to it's k-Nearest Neighbours (kNN) to generate an undirected graph. Fig. 3 shows the resulting graph from a simple 2D example.

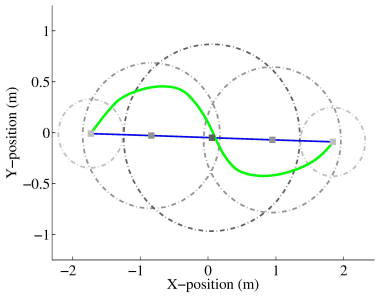

Figure 2: Gaussian ellipses (2σ) around waypoints on a straight line (Blue) between two points in 2D. Demonstrated trajectory shown in green. Larger ellipses (darker) near center have larger covariances.

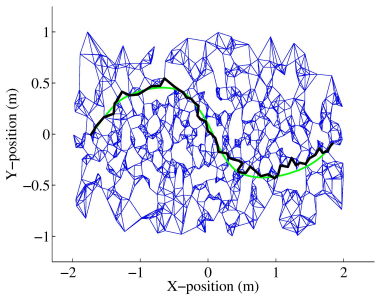

Figure 3: Sampled graph with 1000 nodes and 5NN connected edges (blue) with the projection (black) of demonstrated trajectory (green).

From Continuous Trajectories to Discrete Paths: Discrete MaxEnt IOC aims to find the least biased distribution of paths that matches the feature counts of the demonstrations. For this to be feasible, the demonstrations need to be among the set of all possible paths in the graph. This can be done by *adding* the demonstration to the graph, but this introduces bias. To avoid this, we compute the closest match to the demonstrated trajectory on the graph. For this *projection*, we use a modified Dijkstra's algorithm which finds the shortest path on the graph that is close to the demonstration. This projection does not exactly represent the demonstration, but in practice tends to be a good approximate. Fig.3 shows the demonstrated trajectory and its projection for a simple 2D example.

Discrete MaxEnt IOC: Given the graphs and the projections of demonstrations, we learn a distribution over graph paths by maximizing the probability of the projections under the MaxEnt distribution (Eqn. 2), the learnt parameters being the feature weights (θ):

$$\theta^\star = \arg\max_\theta \mathcal{L}(\mathbf{s}_{1:T}^{demo}) = \arg\max_\theta \frac{e^{-\theta^{\mathrm{T}} \sum_{t=1}^{T} \mathbf{f}(s_t^{demo})}}{\sum_{\mathbf{s}'_{1:T}} e^{-\theta^{\mathrm{T}} \sum_{t=1}^{T} \mathbf{f}(s'_t)}} \tag{3}$$

The features ($f(s_t)$) are task dependent and solely state-based for computational reasons. The optimal weights ($\theta^\star$) are found using gradient based optimizers, making use of efficient softmax inference (Eqn. 1) via an optimized value iteration algorithm for gradient and likelihood computation.

Sampling paths from discrete graph: We can sample paths from our discrete graph in two ways. First, we can probabilistically sample goal-directed paths directly from our learnt path distribution. Alternatively, we can deterministically find the path that minimizes the cost of traversal through the graph ($cost(s_t) = \theta^{\star T} f(s_t)$). In practice, we use rejection sampling to generate paths until they satisfy certain task criteria. These sampled paths typically avoid obstacles and match many of the qualitative properties of demonstrations, but lack the smoothness of demonstrated trajectories due to their discrete nature.

Waypoint-based MaxEnt Inverse Linear-Quadratic Regulation

To better imitate the dynamics of continuous demonstrated trajectories, we construct a continuous MaxEnt IOC model that estimates a distribution of trajectories given a discrete path through our approximation graph. We learn a continuous path distribution through the discrete graphs that match the preferences elicited in the demonstrated trajectories. This allows for us to infer a path through the graph that correlates to the discretely inferred trajectory described previously while retaining the same continous motion patterns of the the demonstrated examples.

LQR model: We form a quadratic cost function, $cost(\mathbf{s}_t, \mathbf{a}_t) = \begin{bmatrix} \mathbf{a}_t \\ \mathbf{s}_t \end{bmatrix}^T \mathbf{\Theta} \begin{bmatrix} \mathbf{a}_t \\ \mathbf{s}_t \end{bmatrix}$, dependent on the state, $\mathbf{s}_t$, and action, $\mathbf{a}_t$, at time t and $\mathbf{\Theta}$ which is the set of learned parameters used to form the cost function. In addition to this state-action cost, we incorporate temporally dependent waypoint penalties which influence our inferred trajectories to align with the discretely inferred path through the graph. The value of a state is dependent on both the cost function and a learned penalty for deviating from an associated waypoint position (s_w), with the association dictated by an indicator function, $\mathbb{I}_{t \in t_w}$. This results in a value function similar to the one in [11]. Here $\mathbf{\Theta}_w$ is the cost parameter matrix for the waypoints, and $\mathbf{\Theta}_f$ is the cost parameter matrix for the final state where the deviation from the expected final waypoint, s_G, has a unique penalty.

$$Q(\boldsymbol{s}_t, \boldsymbol{a}_t) \triangleq cost(\mathbf{s}_t, \mathbf{a}_t) + \\ \mathop{\mathbb{E}}_{P(s_{t+1}|a_t,s_t)} \left[\begin{cases} \mathop{\mathbb{I}}_{t+1 \in t_w} (\boldsymbol{s}_{t+1} - \boldsymbol{s}_{w_{t+1}})^T \mathbf{\Theta}_w (\boldsymbol{s}_{t+1} - \boldsymbol{s}_{w_{t+1}}) + \text{softmax}(Q(\boldsymbol{s}_{t+1}, \boldsymbol{a}_{t+1})), & t+1 < T \\ (\boldsymbol{s}_{t+1} - \boldsymbol{s}_G)^T \mathbf{\Theta}_f (\boldsymbol{s}_{t+1} - \boldsymbol{s}_G)], & t+1 = T \end{cases} \right]$$

The path's sequence of graph states serve as waypoints in our model with a learned quadratic term that penalizes deviation from the graph states. Sampling from this trajectory distribution projects our discrete paths back into the continuous space by smoothing the resulting discrete graph path. MaxEnt IOC includes uncertainty estimates in its probabilistic predictions. This suggests that, should an available computational budget allow for it, multiple paths in the discrete representation can be obtained based on the MaxEnt IOC distribution, smoothed to provide continuous trajectories, and then the best of those selected.

Learning Motion Preferences: The parameters of the above model can then be trained on a set of demonstrated trajectories and an associated set of discrete waypoints for each trajectory. This forms

a convex optimization problem where the gradients are formed via expectation matching:

$$\nabla L(\mathbf{\Theta}) = \mathbb{E}_{\hat{\pi}}\left[\sum_{t=1}^{T-1}\begin{bmatrix}\mathbf{a}_t\\ \mathbf{s}_t\end{bmatrix}\begin{bmatrix}\mathbf{a}_t\\ \mathbf{s}_t\end{bmatrix}^T\right] - \mathbb{E}_{\tilde{\pi}}\left[\sum_{t=1}^{T-1}\begin{bmatrix}\mathbf{a}_t\\ \mathbf{s}_t\end{bmatrix}\begin{bmatrix}\mathbf{a}_t\\ \mathbf{s}_t\end{bmatrix}^T\right],$$

$$\nabla L(\mathbf{\Theta}_w) = \mathbb{E}_{\hat{\pi}}\left[\sum_{t\in t_w}(\mathbf{s}_t - \mathbf{s}_w)(\mathbf{s}_t - \mathbf{s}_w)^T\right] - \mathbb{E}_{\tilde{\pi}}\left[\sum_{t\in t_w}(\mathbf{s}_t - \mathbf{s}_w)(\mathbf{s}_t - \mathbf{s}_w)^T\right],$$

$$\nabla L(\mathbf{\Theta}_f) = \mathbb{E}_{\hat{\pi}}\left[(\mathbf{s}_T - \mathbf{s}_G)(\mathbf{s}_T - \mathbf{s}_G)^T\right] - \mathbb{E}_{\tilde{\pi}}\left[(\mathbf{s}_T - \mathbf{s}_G)(\mathbf{s}_T - \mathbf{s}_G)^T\right].$$

From Discrete Paths to Continuous Trajectories: We generate samples from our estimated probability distribution in a two-step process. We first sample a set of paths through our sparse, discrete graph representation. The graph nodes of this path then serve as waypoints for our continuous trajectory distribution estimate. A set of trajectory samples are obtained by conditioning on the graph path sample. The fitness of these sampled trajectories are evaluated by considering aspects of the manipulation task that were ignored or only partially incorporated in our abstract representations. In practice, LQR trajectory samples tend to be much smoother than our discrete paths and can directly be executed on the manipulator.

Evaluation

Experimental setup and Data Collection

We evaluate our approach on three separate tasks with a 7-DOF Barrett Whole Arm Manipulator (BarrettWAM). The tasks involve the arm moving in a specific manner through a tabletop scene to reach a target object while avoiding obstacles. The tasks are:

- $\mathcal{T}_1$. Approach the target object from the right side (from the arm's perspective)
- $\mathcal{T}_2$. Approach the target object from the left side
- $\mathcal{T}_3$. Carry a liquid filled can to a target destination without spilling

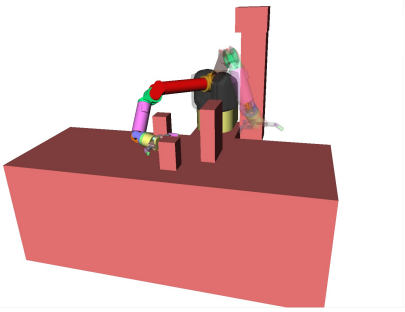

Figure 4: BarrettWAM and three objects with Start (transparent) and Goal (solid) configurations from the *Approach-Right* ($\mathcal{T}_1$) task. The target is the object closest to the end-effector.

For each task, we collected demonstrations from four different users. Half had never used the robotic platform while the rest had some experience with it. For each task, we collected 16 demonstrations from each user with different start and goal configurations and different obstacle locations. We recorded the joint angles of the demonstrated trajectories as well as the obstacle positions in each case. We use the OpenRAVE [18] virtual environment for testing our system as it provides good primitives for manipulator kinematics, geometry and visualization. Fig. 4 shows an OpenRAVE render of a tabletop scene from one of our datasets.

Cost function features

We use the following state-based features ($f(s_t)$) to characterize trajectories in our lower-dimensional representation:

1. Distance from robot to objects (min, average, target distance from end-effector, etc.);
2. Histogram over distance to objects;
3. Self-collision distances and corresponding histograms;
4. Histograms of position differences between end effector and target, right/left flags; and
5. Histograms over difference in end-effector orientation from start configuration

and additional features based on the start, goal and target object and a constant feature. In total, we have 95 features for all of our tasks, a majority being integer or binary features and the rest individually scaled to be between 0-1.

For linear-quadratic regulation (LQR) we define the state, $\mathbf{s}_t = [\phi_t^1, ..., \phi_t^7, \dot{\phi}_t^1, ..., \dot{\phi}_t^7, \ddot{\phi}_t^1, ..., \ddot{\phi}_t^7, 1]^T$, and action, $\mathbf{a}_t = [\dddot{\phi}_t^1, ..., \dddot{\phi}_t^7]^T$, at time t where $(\phi^1, ..., \phi^7)$ denotes the joint angles, $(\dot{\phi}^1, ..., \dot{\phi}^7)$ the joint velocities, and $(\ddot{\phi}^1, ..., \ddot{\phi}^7)$ the joint accelerations. A unit constant is added to the state representation to incorporate linear features into the quadratic cost function formulation.

Task		Collision-Free (%)	Completes Task (%)	Overall Success (%)
Approach-Right	Test	93	96	90
	Random	96	99	96
Approach-Left	Test	98	91	88
	Random	90	93	86
Can-Moving	Test	92	70	66
	Random	95	58	58

Table 1: Learning performance on the withheld set (*Test*) and the randomly generated set (*Random*).

Evaluation measures

We evaluate the suitability of trajectories generated from our approach by comparing them against the *true* demonstrations. Learned trajectories need to be collision-free. Additionally, they need to satisfy task-dependent metrics. For the *Can-Moving* task, we require that the end-effector's maximum deviation in pitch and roll from the start configuration be less than that of the demonstration. For the *Approach* tasks, we compute the percentage of the final 25% of the demonstration that stays on the desired side of the target. We require that the learned trajectory match or exceed this.

Additionally, we test on randomly generated scenes. Each scene has a varying number of randomly generated box shaped objects (random number and size) in random configurations. The target object, the start and goal configuration of the robot are also chosen randomly (subject to task constraints). The metrics are similar for the random tests, but fixed apriori.

Results

For each task, we trained our system using 70% of the demonstrations and test it on the rest. We also generated 20 random scenes per run. In all tests, the discrete graph had 210000 nodes with 15NN edges. We trained the LQR model on the fly using least-cost graph trajectories post learning and the training demonstrations. Table 1 shows the performance from our learned LQR model, averaged over six random trials.

On the two *Approach* tasks the algorithm performs very well, avoiding obstacles and successfully completing the task on more than 85% of the scenes. The algorithm also generalizes well on random scenarios. For the *Can-Moving* task, the algorithm learns to avoid obstacles but fails to satisfy the task metric in nearly 30% of the scenes. Performance on random scenes is slightly poor, but overall, the results are highly encouraging. One reason for reduced performance on the *Can-Moving* task could be the sparseness of the graph and projections, leading to a poor representation of the small pitch and roll changes in the demonstrations, ultimately resulting in learned distributions that do not satisfy the maximum orientation difference metric. Sampling many paths and LQR smoothing improves results significantly (66% with LQR to 45% without), but falls short of fixing it completely.

Discussion

We presented a two stage algorithm for Maximum Entropy Inverse Optimal Control and applied it to three test scenarios with a 7DOF robot manipulator. The first stage is a discrete graph based representation used to learn a distribution of paths matching projections of demonstrations. The second stage generates a continuous trajectory from a sample graph path via a waypoint based LQR formulation. The final trajectory is evaluated against a held-out set of user demonstrations and random tests; showing that the algorithm is capable of generating human-like trajectories.

The key strengths and weaknesses of the approach lie in the discrete graph representation. It has many merits; representing the space as a graph preserves the theoretical guarantees of MaxEnt IOC, permits the use of optimized value iteration for inference, and results in better generalization. The downsides are the sparseness of the graph, inexact representation of the demonstrations and exponential complexity with increasing problem dimension. Clever sampling and projection techniques can alleviate these to a certain extent, but it is hard to beat the curse of dimensionality. Additionally, generating a solution at test-time involves generating a new graph, which is time-consuming.

While current results are promising, a more thorough evaluation is needed. A good way to measure performance would be to conduct user studies comparing the learned trajectories with human-demonstrations for the same task. The user could be allowed to choose the more *human-like* trajectory, establishing a clear success metric. Additionally, the algorithm needs to be tested on harder and varied manipulation tasks. Also, tests comparing performance with respect to graph sparsity, projection error and different feature databases should give good insights. Implementing better graph generation and projection schemes should improve performance and are clear areas for future work.

Acknowledgments

This work was funded in part by the National Science Foundation under contract NSR-NRI 1227234 and Grant No: 1227495, Purposeful Prediction: Co-robot Interaction via Understanding Intent and Goals.

References

[1] Dragan, A.D., Lee, K.C., Srinivasa, S.S.: Legibility and predictability of robot motion. In: Human-Robot Interaction (HRI), 2013 8th ACM/IEEE International Conference on, IEEE (2013) 301–308

[2] Jain, A., Wojcik, B., Joachims, T., Saxena, A.: Learning trajectory preferences for manipulators via iterative improvement. In: Advances in Neural Information Processing Systems. (2013) 575–583

[3] Zucker, M., Ratliff, N., Dragan, A.D., Pivtoraiko, M., Klingensmith, M., Dellin, C.M., Bagnell, J.A., Srinivasa, S.S.: Chomp: Covariant hamiltonian optimization for motion planning. The International Journal of Robotics Research **32**(9-10) (2013) 1164–1193

[4] Abbeel, P., Ng, A.Y.: Apprenticeship learning via inverse reinforcement learning. In: Proceedings of the International Conference on Machine learning, ACM (2004) 1

[5] Ziebart, B.D., Maas, A.L., Bagnell, J.A., Dey, A.K.: Maximum entropy inverse reinforcement learning. In: AAAI. (2008) 1433–1438

[6] Ng, A.Y., Russell, S.J.: Algorithms for inverse reinforcement learning. In: Proceedings of the International Conference on Machine Learning

[7] Ratliff, N.D., Silver, D., Bagnell, J.A.: Learning to search: Functional gradient techniques for imitation learning. Autonomous Robots **27**(1) (2009) 25–53

[8] Levine, S., Koltun, V.: Continuous inverse optimal control with locally optimal examples. arXiv preprint arXiv:1206.4617 (2012)

[9] Abbeel, P., Coates, A., Quigley, M., Ng, A.Y.: An application of reinforcement learning to aerobatic helicopter flight. Advances in neural information processing systems **19** (2007) 1

[10] Henry, P., Vollmer, C., Ferris, B., Fox, D.: Learning to navigate through crowded environments. In: Robotics and Automation (ICRA), 2010 IEEE International Conference on, IEEE (2010) 981–986

[11] Ziebart, B., Dey, A., Bagnell, J.A.: Probabilistic pointing target prediction via inverse optimal control. In: Proceedings of the 2012 ACM international conference on Intelligent User Interfaces, ACM (2012) 1–10

[12] Topsøe, F.: Information theoretical optimization techniques. Kybernetika **15**(1) (1979) 8–27

[13] Grünwald, P.D., Dawid, A.P.: Game theory, maximum entropy, minimum discrepancy, and robust Bayesian decision theory. Annals of Statistics **32** (2004) 1367–1433

[14] Ziebart, B.D.: Modeling purposeful adaptive behavior with the principle of maximum causal entropy. (2010)

[15] Kalakrishnan, M., Chitta, S., Theodorou, E., Pastor, P., Schaal, S.: Stomp: Stochastic trajectory optimization for motion planning, IEEE (2011) 4569–4574

[16] Schulman, J., Lee, A., Awwal, I., Bradlow, H., Abbeel, P.: Finding locally optimal, collision-free trajectories with sequential convex optimization. RSS (2013)

[17] Dudik, M., Phillips, S.J., Schapire, R.E.: Performance guarantees for regularized maximum entropy density estimation. In: Learning Theory. Springer (2004) 472–486

[18] Diankov, R., Kuffner, J.: Openrave: A planning architecture for autonomous robotics. (2008)