Learning Human Utility from Video Demonstrations for Deductive Planning in Robotics

Nishant Shukla, Yunzhong He, Frank Chen, and Song-Chun Zhu
Center for Vision, Cognition, Learning, and Autonomy
University of California, Los Angeles, United States
{nxs,hyz331,frank.chen}@ucla.edu, sczhu@stat.ucla.edu

Abstract: We uncouple three components of autonomous behavior (utilitarian value, causal reasoning, and fine motion control) to design an interpretable model of tasks from video demonstrations. Utilitarian value is learned from aggregating human preferences to understand the implicit goal of a task, explaining *why* an action sequence was performed. Causal reasoning is seeded from observations and grows from robot experiences to explain *how* to deductively accomplish subgoals. And lastly, fine motion control describes *what* actuators to move. In our experiments, a robot learns how to fold t-shirts from visual demonstrations, and proposes a plan (by answering *why*, *how*, and *what*) when folding never-before-seen articles of clothing.

Keywords: Hierarchical planning, Utility, Deductive learning, Explainable AI

1 Introduction

Explicitly programming service robots to accomplish new tasks in uncontrolled environments is time-consuming, error-prone, and sometimes even infeasible. In Learning from Demonstration (LfD), many statistical models have been proposed that maximize the likelihood of observations [1]. For example, Bayesian formulations [2] assume a prior model of the goal, and use Bayes' Theorem to explain the relationship between the posterior and likelihood. These Bayesian formulations learn a model of the demonstrated task most consistent with training data. Such approaches are often referred to as *inductive learning* [3].

In contrast, robot autonomy was originally studied as a rule-based deductive learning system [4, 5]. There is a paradigm shift in applying inductive models to *deduction* based inference. In this paper, we explore a middle-ground, where deductive rules are learned through statistical techniques.

Specifically, we teach a robot how to fold shirts through human demonstrations, and have it reproduce the skill under both different articles of clothing and different sets of available actions. Our experimental results show good performance on a two-armed industrial robot following causal chains that maximize a learned latent utility function. Most importantly, the robot's decisions are interpretable, facilitating immediate natural language description of plans [6].

Human preferences are modeled by a latent utility function over the states of the world. To rank preferences, we pursue relevant fluents of a task, and then learn a utility function based on these fluents. For example, Figure 1 shows the utility landscape for a cloth-folding task, obtained through 45 visual demonstrations.

The utility landscape shows a global perspective of candidate goal states. To close the loop with autonomous behaviour, we further design a dynamics equation to connect high-level reasoning to low-level motion control. The primary contributions of our work include:

1. Learning an *interpretable* utility of continuous states, independent of system dynamics.
2. *Deductively* exploring goal-reachability under different available actions.
3. Proposing *"Fluent Dynamics"* to bridge low-level motion trajectory with high-level utility.
4. Teaching a robot to fold t-shirts, and have it *generalize* to arbitrary articles of clothing.

1st Conference on Robot Learning (CoRL 2017), Mountain View, United States.

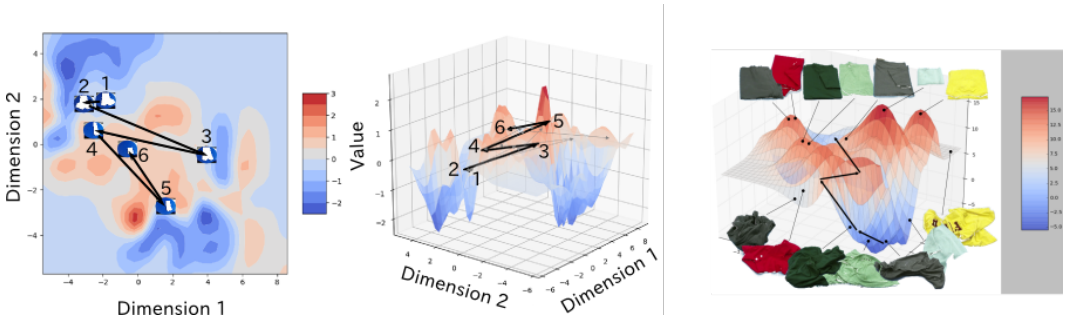

Figure 1: The utility landscape identifies desired states. This one, in particular, is trained from 45 cloth-folding video demonstrations. For visualization purposes, we reduce the state-space to two dimensions through multidimensional scaling (MDS). The canyons in this landscape represent wrinkled clothes, whereas the peaks represent well-folded clothes. Given this learned utility function, a robot chooses from an available set of actions to craft a motion trajectory that maximizes its utility.

2 Related Work

Modeling and Learning Human Utility: Building computational models for human utilities could be traced back to the English philosopher, Jeremy Bentham, in his works on ethics known as utilitarianism [7]. Utilities, or values, are also used in planning schemes like Markov decision process (MDP) [8], and are often associated with states of a task. However, in the literature of MDP, the "value" is not a reflection of true human preference and, inconveniently, is tightly dependent on the agent's actions.

Zhu et al. [9] first modeled human utilities over physical forces on tools, and proposed effective algorithms to learn utilities from videos. Our work differs in 4 ways: 1) we generalize the linear SVM separator ("rankSVM") so that the utility of each individual fluent is dictated not by a "weight" but instead a non-linear utility function; 2) relevant fluents are pursued one-by-one from a large dictionary; 3) we learn from external fluents, such as states of the world, instead of internal fluents, such as states of the agent; 4) the utility function drives robot motion.

Inverse Reinforcement Learning: Inverse reinforcement learning (IRL) aims to determine the reward function being locally optimized from observed behaviors of the actors [10]. In the IRL framework, we are satisfied when a robot mimics observed action sequences, maximizing the likelihood. Hadfieldmenell et al. [11] defined cooperative inverse reinforcement learning (CIRL), which allows reward learning when the observed behaviors could be sub-optimal, based on human-robot interactions. In contrast to IRL or CIRL, our method does not aim to reproduce action sequences, nor is our approach dependent on the set of possible actions. It avoids the correspondence problem in human-robot knowledge transfer by learning the global utility function over observed states, rather than learning the local reward function from actions directly. Furthermore, our work avoids the troublesome limitations of subscribing to a Markov assumption [12, 13].

Robot Learning from Demonstrations: Learning how to perform a task from human demonstrations has been a challenging problem for artificial intelligence and robotics, and various methods were developed trying to solve the problem [14]. Learning the task of cloth-folding from human demonstrations, in particular, has been studied before by Xiong et al. [15]. While most of the existing approaches focus on reproducing the demonstrator's action sequence, our work tries to model human utilities from observations, and generates task plans deductively from utilities.

3 Model

Definition 1. Environment: The world (or *environment*) is defined by a generative composition model of objects, actions, and changes in conditions [16]. Specifically, we use the stochastic context-free And-Or graph (AOG), which explicitly models variations and compositions of spatial (S), temporal (T), and causal (C) concepts, called the STC-AOG [15].

The atomic (*terminal*) units of this composition grammar are tuples of the form $(F_{start}, u_{[1:t]}, F_{end})$, where F_{start} and F_{end} are pre- and post-fluents of a sequence of interactions $u_{[1:t]}$. Concretely, the sequence of interactions $u_{[1:t]}$ is implemented by spatial and temporal features of human-object interactions (4D HOI) [17]. See definition 9.

Definition 2. State: A *state* is a configuration of the believed model of the world. In our case, a state is a parse-graph (pg) of the And-Or graph, representing a selection of parameters (Θ_{OR}) for each Or-node. The set of all parse-graphs is denoted Ω_{pg}.

Definition 3. Fluent: A *fluent* is a condition of a state that can change over time [18]. It is represented as a real-valued function on the state (indexed by $i \in \mathbb{N}$): $f_i : \Omega_{pg} \to \mathbb{R}$.

Definition 4. Fluent-vector: A *fluent-vector* F is a column-vector of fluents: $F = (f_1, f_2, ..., f_k)^\intercal$.

Definition 5. Goal: The *goal* of a task is characterized by a fluent-change $\triangle F$. The purpose of learning the utility function is to identify reasonable goals.

3.1 Utility Model

We assume human preferences are derived from a utilitarian model, in which a latent utility function assigns a real-number to each configuration of the world. For example, if a state pg^1 has a higher utility than another state pg^2, then the corresponding ranking is denoted $pg^1 \succ pg^2$, implying the utility of pg^1 is greater than the utility of pg^2.

Each video demonstration contains a sequence of n states $pg^0, pg^1, ..., pg^n$, which offers $\binom{n}{2} = n(n-1)/2$ possible ordered pairs (*ranking constraints*). Given some ranking constraints, we define an energy function by how consistent a utility function is with the constraints.

The energy function described above is used to design its corresponding Gibbs distribution. In the case of Zhu and Mumford [19], a maximum entropy model reproduces the marginal distributions of fluents. Instead of matching statistics of observations, our work attempts to model human preferences. We instead use a maximum margin formulation, and select relevant fluents by minimizing the ranking violations of the model. The specific details of this preference model is described below.

3.2 Minimum Violations

Let $\mathcal{D} = \{f^{(1)}, f^{(2)}, ...\}$ be a dictionary of fluents, each with a latent utility function $\lambda : \mathbb{R} \to \mathbb{R}$. Using a sparse coding model, the utility of a parse-graph pg is estimated by a small subset of relevant fluents $F = \{f^{(1)}, f^{(2)}, ..., f^{(K)}\} \subset \mathcal{D}$. Denote $\Lambda = \{\lambda^{(1)}(), \lambda^{(2)}(), ..., \lambda^{(K)}()\}$ as the corresponding set of utility functions for each fluent in F. For example, 12 utility functions learned from human preferences are shown in Figure 2, approximated by piecewise linear functions. The total utility function is thus,

$$U(pg; \Lambda, F) = \sum_{\alpha=1}^{K} \lambda^{(\alpha)}(f^{(\alpha)}(pg)) \tag{1}$$

Of all selection of parameters (Λ) and fluent-vectors (F) that satisfy the ranking constraints, we choose the model with minimum ranking violations. In order to learn each utility function in Λ, we treat the space of fluents as a set of *alternatives* [20]. Let R denote the set of rankings over the alternatives. Each human demonstration is seen as a ranking $\sigma_i \in R$ over the alternatives. We say $a \succ_{\sigma_i} b$ if person i prefers alternative a to alternative b. The collection of a person's rankings is called their preference profile, denoted $\vec{\sigma}$.

Each video v provides a preference profile $\vec{\sigma}_v$. For example, we assume at least the following ranking: $pg^* \succ_{\sigma_v} pg^0$, where pg^0 is the initial state and pg^* is the final state. The learned utility functions try to satisfy $U(pg^*) > U(pg^0)$.

U is treated as a ranking score: higher values correspond to more favorable states. We want to model the goal of a task using rankings obtained from visual demonstrations. The goal model, or *preference model*, of a parse-graph pg takes the Gibbs distribution of the form, $\mathbf{p}(pg; \Lambda, F) = \frac{1}{Z}\mathbf{e}^{U(pg;\Lambda,F)}$,

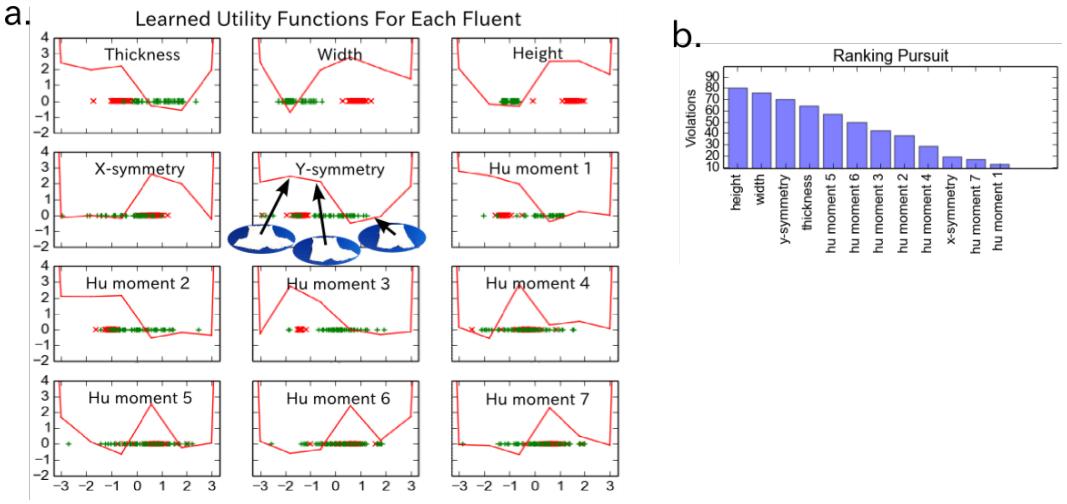

Figure 2: **(a)** The 12 curves represent the negative utility function ($-\lambda$) corresponding to each fluent. The functions are negated to draw parallels with the concept of potential energy. Red marks indicate fluent values of pg^0, which the learned model appears to avoid, and the green marks indicate fluent values of the goal pg^*, which the learned model appears to favor. Notice how the y-symmetry potential energy decreases as the cloth becomes more and more symmetric. By tracing the change in utilities of each individual fluent, the robot can more clearly explain *why* it favors one state over another. **(b)** The ranking pursuit algorithm extracts fluents greedily to minimize ranking violations. As shown in the chart, the top 3 most important fluents for the task of cloth-folding are height, width, and y-symmetry.

where $U(\ \cdot\ ; \Lambda, F)$ is the total utility function that minimizes ranking violations:

$$\begin{aligned} \min \sum_{\alpha=1}^{K} \int_x \lambda''^{(\alpha)} dx + C \sum_v \xi_v \\ \text{s.t.} \sum_\alpha \left(\lambda^{(\alpha)}(f^{(\alpha)}(pg_v^*)) - \lambda^{(\alpha)}(f^{(\alpha)}(pg_v^0))\right) \\ > 1 - \xi_v, \\ \xi_v \geq 0. \end{aligned} \tag{2}$$

Here, ξ_v is a non-negative slack variable analogous to margin maximization. C is a hyper-parameter that balances the violations against smoothness of the utility functions [21]. Of all utility functions, we select the one which minimizes the ranking violations. The next section explains how to select the optimal subset of fluents.

3.3 Ranking pursuit

The empirical rankings of states $pg^* \succ pg^0$ in the observations must match the predicted ranking. We start with an empty set of fluents $F = \{\}$, and select from the elements of $\mathcal{D}$ that result in the least number of ranking violations.

This process continues greedily until the amount of violations can no longer be substantially reduced. Figure 2 shows empirical results of pursuing relevant fluents for the cloth-folding task. The dictionary of initial fluents may be hand-designed or automatically learned through statistical means, such as from hidden layers of a convolutonal neural network.

3.4 Ranking Sparsity

The number of ranking pairs we can extract from the training dataset is not immediately obvious. For example, each video demonstration supplies ordered pairs of states that we can use to learn a utility function. A sequence of n states (pg^0, pg^1, ..., pg^n) allows $\binom{n}{2} = n(n-1)/2$ ordered pairs.

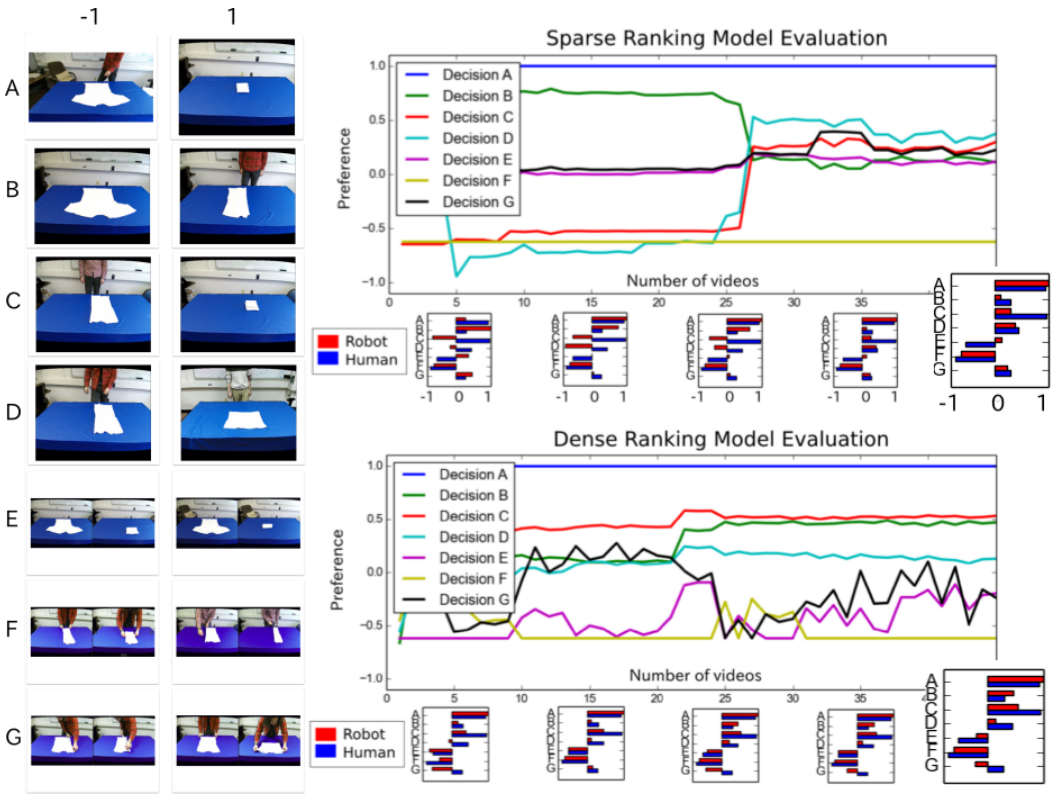

Figure 3: The sparse and dense ranking models are evaluated by how quickly they converge and how strongly they match human preferences. The x-axis on each plot indicates the number of unique videos shown to the learning algorithm. The y-axis indicates two alternatives (1 vs. -1) for 7 decisions (A, B, C, D, E, F, and G) of varying difficulty. The horizontal bar-charts below each plot show comparisons between human and robot preferences. As more videos are made available, both models improve performance in convergence as well as alignment to human preferences (from 330 survey results).

On one end of the spectrum, which we call *sparse ranking*, we know at the very least that $pg^n \succ pg^0$ for each demonstration. This is a safe bet since each video demonstration is assumed to successfully accomplish the goal. However, the utility model throws out useful information when ignoring the intermediate states.

On the other end, in *dense ranking*, all $\binom{n}{2}$ are used. Despite using all information available, this approach may be prone to introducing many ranking violations.

Figure 3 visualizes performance of both approaches as we increment the number of available video demonstrations.

4 Utility-Driven Task Planning

The learned utility function is used to drive robot behavior. In order to perform low-level motion control in the real world, the robot needs a concept of what actions are possible in the observed state. First, we explicitly define the concept of relevant fluents, and then use it to describe the preconditions of an action.

Definition 6. Relevant fluent: A fluent-vector might contain irrelevant fluents for an action. The *relevant fluents of an action* are a subset of a fluent-vector, described using an element-wise multiplication of a binary vector w by the fluent-vector, $w \circ F = WF$, where W is a diagonal matrix.

Definition 7. Action: An *action* sequence $u_{[1:t]}$ causes a change in fluents given a precondition. The precondition of an action depends on relevant fluents W_u as well as a representative example pg_u. Let $\theta = (W_u, pg_u)$ denote these parameters. The precondition is a probability $P(X = pg \mid u;\ \theta_u)$ over a random variable $X \in \Omega_{pg}$.

A tuple of (F_{start}, $u_{[1:t]}$, F_{end}) is used to construct the compositional model of the environment. The energy of a pg is computed by comparing the difference between the relevant fluents of the observation Wpg to the relevant fluents of the action Wpg_u, Cost$(pg;\ W_u, pg_u) = ||W_u(pg - pg_u)||$.

4.1 Representing what, how, and why

Fluents provide information on the utility of the state. Therefore, a robot can identify a fluent-change $\triangle F$ to maximize its utility, explaining *why* it acts. Figuring out how to achieve the fluent-change requires the robot to accomplish a sequence of interactions between itself and the environment, which we call the *union space*. Interacting with the environment requires the robot to be aware of the span of its own actuators, called the *actuator space*, where it identifies *what* joints move.

The three spaces are tightly coupled, and can be used to perform real-time robot executions. We explain each space separately, and then provide a unified dynamics formulation for robot task planning.

Definition 8. Actuator Space: We characterize a robot actuator by its η degrees-of-freedom. The actuator space Ω_A is a set of all valid η-dimensional vectors. At any point in time, a robot can be represented as a point in this space, $a \in \Omega_A \subset \mathbb{R}^\eta$.

For example, our robot platform is represented by a 16-dimensional vector, since each arm has 7 joints and an open/close grasp-status for each hand. As the robot moves in the real-world, this 16-dimensional point drifts in the actuator space.

Definition 9. Union Space: The interactions between an agent and object are jointly interpreted in what we call the union space Ω_U. The distance between an agent's end-effector and the midpoint of an object is one such example. These computed values depend of the current actuator position, $u(a) \in \Omega_U$. The table below shows a possible 12-dimensional vector in the union space.

#	Feature	Agent	Object
1	distance	left end-effector	cloth
2-5	θ_x, θ_y, θ_z, θ_w	left end-effector	cloth
6	grasp status	left end-effector	N/A
7	distance	right end-effector	cloth
8-11	θ_x, θ_y, θ_z, θ_w	right end-effector	cloth
12	grasp status	right end-effector	N/A

Examples of rows in this table include, but are not limited to, distance and orientation between an end-effector and an object. Clearly, a robot adjusting its position in the actuator space affects its position in the union space. A sequence of such vectors in the union space causes a fluent change. Inferring how fluents change from a trajectory in the union space is given as a hierarchical task plan (see Definition 1) by the domain expert. Further causal relationships are gathered through exploration, but a deeper investigation in learning causality is beyond the scope of this paper.

Definition 10. Fluent Space: A fluent space Ω_F is the set of all possible fluent vectors F. A sequence of vectors from the union space $u_{[1:t]}$ causes the value of the fluent vector to change, $F(u_{[1:t]}) \in \Omega_F$.

4.2 Fluent Dynamics

Influenced by previous work [22] in specifying robot behavior independently of its actuators, this paper proposes a control formulation to unify low-level mechanics with high-level preference. In deductive planning, the optimal selection of actions achieves a goal with minimum cost. We define the cost of a sequence of actions $V(a_{[1:t]})$ by the utility of the resulting fluent-vector. With that in mind, the robot must use its available actionable information to maximize utility, $a^*_{[1:t]} = \arg\max_{a_{[1:t]}} V(a_{[1:t]})$. Optimal action sequences will satisfy $\partial V / \partial a_{[1:t]} = 0$. The gra-

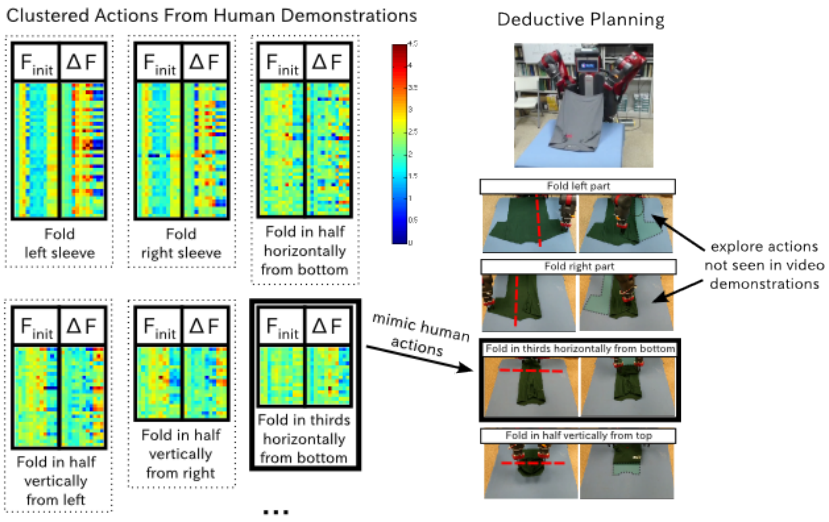

Figure 4: After clustering fluent-changes from the training videos, 11 actions are automatically captured. Each action is shown by 2 matrices: F_{init} and $\triangle F$. The rows of the matrix correspond to a concrete example. The columns of the matrix correspond to the various fluents. The robot can understand the relevant fluents associated per each action.

dient of V with respect to $a_{[1:t]}$ can be computed using the chain rule,

$$\frac{\partial V}{\partial a_{[1:t]}} = \frac{\partial V}{\partial F} \frac{\partial F}{\partial u_{[1:t]}} \frac{\partial u_{[1:t]}}{\partial a_{[1:t]}} \tag{3}$$

The first factor $\partial V / \partial F$ comes immediately from the utility learning section of this paper. The second factor $\partial F / \partial u_{[1:t]}$ is solved using the assumed And-Or compositional model of the world, as explain in Definition 1. And lastly, inverse kinematics and optimal control methods directly solve $\partial u_{[1:t]} / \partial a_{[1:t]}$ [22].

The trajectory Γ between two robot states is $\arg\min_{a_{[1:t]}} \sum_{a_{[1:t]}} |V(a_{[1:t]}) \cdot \triangle a_{[1:t]}|$ [23].

The optimal trajectory of actions to achieve the highest value is estimated using beam-search and dynamic programming. Sub-goal reachability is automatically solved through the *Confident Execution* algorithm in [24] as shown in Figure 4.

5 Implementation and Experimental Results

We learn our utility function from 45 RGB-D (*pointcloud*) video demonstrations of t-shirt folding. This dataset splits into 30 for training and 15 for testing. The videos were recorded on a separate Kinect camera at a different orientation in a separate setting by different people.

At each frame, the vision processing step segments the cloth using graph-based techniques on the 2D RGB image [25, 26], and then the extracted 3D cloth pointcloud is aligned to its principal axis. Next, we extract fluents from the pointcloud being tracked. Examples of a couple fluents include width, height, thickness, x-symmetry, y-symmetry, and the 7 moment invariants [27]. Width, height, and thickness are calculated in meters, after aligning the segmented pointcloud to its principal axis. X-, and y-symmetry scores are calculated by measuring the symmetric difference of folding the segmented pointcloud on the first two principle axes. The Hu-moments are calculated on the binary mask of the cloth.

We extract both automatic and hand-designed fluents to obtain a training dataset $\mathcal{X}$ to pursue relevant fluents and obtain an optimal ranking. Each utility function is approximated by a piecewise linear

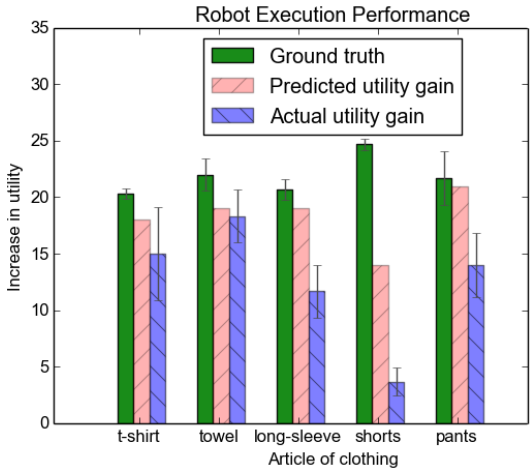

Figure 5: Robot task execution is evaluated in two ways. First, we measure how well the robot can predict the increase in utility through deductive reasoning compared to ground truth by having a human perform the same action. Second, we compare the prediction to actual utility gain after executing the action sequence using Fluent Dynamics. Our experiments show strong generalizability to other articles of clothing, even though the training videos only contained t-shirt folding demonstrations. As shown above, the robot can detect execution failure when folding shorts by detection an anomaly in actual vs. predicted utility gain.

function. The vision processing code, ranking pursuit algorithm, and the RGB-D dataset of cloth-folding are open-sourced on the author's website [1].

We cross-validate the fitting of the learned utility model to ensure generalizability to the other training videos. Furthermore, we perform external evaluations on 330 individuals to compare how well the learned preference model matches human judgement. In this survey, each human was asked to make a decision on 7 choices after being told, "A robot attempts to fold your clothes. Of each outcome, which do you prefer?"

The qualitative comparisons between human and robot preferences is shown in figure 3. Overall, our model quickly converges to human preferences (within 27 videos in the sparse ranking model and just 5 videos in the dense ranking model).

To further evaluate how effectively the learned utility function assists in deductive planning, we conduct a series of experiments on a robot platform. We solve the three factors in the dynamics equation independently. The learned utility function gives us the first factor $\partial V/\partial F$, the STC-AOG units provide the second factor $\partial F/\partial a_{u_{[1:t]}}$, and an off-the-shelf inverse kinematics library solves $\partial u_{[1:t]}/\partial a_{[1:t]}$.

In the experiments, the robot is presented with a never-before-seen article of clothing, and asked to identify a goal, plan an action sequence to achieve that goal, predict the utility gain, and then perform the plan to compare against actual utility gain. Figure 5 shows how well the robot's performance matches both its own predictions as well as the ground truth.

6 Conclusions and Future Work

The learned utility model strongly matches preferences of 330 human decisions, capturing the commonsense goal. The inferred action plan is not only interpretable (*why*, *how*, and *what*), but also performable on a robot platform to complete the learned task in a completely new situation. In future work, we would like to incorporate social choice theory [28], understand inconsistencies between human preferences [29], and learn fluents automatically using neural techniques [30].

[1] https://github.com/BinRoot/Fluent-Extractor

Acknowledgments

This work is supported by DARPA-XAI Project Award No. N66001-17-2-4029, MURI ONR N00014-16-1-2007, and DARPA FunLoL project W911NF-16-1-0579. We would also like to explicitly thank Yixin Zhu, Mark Edmonds, Hangxin Liu, and Feng Gao for their help with the robot platform.

References

[1] S. Chernova and A. L. Thomaz. Robot learning from human teachers. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 8(3):1–121, 2014.

[2] V. N. Vapnik and V. Vapnik. *Statistical learning theory*, volume 1. Wiley New York, 1998.

[3] T. L. Griffiths, C. Kemp, and J. B. Tenenbaum. Bayesian models of cognition. 2008.

[4] C. Boutilier, R. Dearden, and M. Goldszmidt. Exploiting structure in policy construction. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'95, pages 1104–1111, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc. ISBN 1-55860-363-8. URL http://dl.acm.org/citation.cfm?id=1643031.1643043.

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

[6] C. Liu, S. Yang, S. Saba-Sadiya, N. Shukla, Y. He, S. Zhu, and J. Y. Chai. Jointly learning grounded task structures from language instruction and visual demonstration. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 1482–1492, 2016.

[7] J. Bentham. *An Introduction to the Principles of Morals and Legislation*. 1789.

[8] M. L. Puterman. Markov decision process. *Journal of the Royal Statistical Society*, 158(3): 1–16, 1994.

[9] Y. Zhu, C. Jiang, Y. Zhao, D. Terzopoulos, and S. C. Zhu. Inferring forces and learning human utilities from videos. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3823–3833, 2016.

[10] S. R. Andrew Y. Ng. Algorithms for inverse reinforcement learning. In *Proceedings of International Conference on Machine Learning (ICML 2000)*, Stanford, USA, June 2000.

[11] D. Hadfieldmenell, A. Dragan, P. Abbeel, and S. Russell. Cooperative inverse reinforcement learning. 2016.

[12] A. Gabaldon. Non-markovian control in the situation calculus. *Artificial Intelligence*, 175(1): 25–48, 2011.

[13] S. Thiébaux, C. Gretton, J. Slaney, D. Price, and F. Kabanza. Decision-theoretic planning with non-markovian rewards. *Journal of Artificial Intelligence Research*, 25:17–74, 2006.

[14] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[15] C. Xiong, N. Shukla, W. Xiong, and S. Zhu. Robot learning with a spatial, temporal, and causal and-or graph. In *Proceedings of International Conference on on Robotics and Automation (ICRA 2016)*, Stockholm, Sweden, May 2016.

[16] N. Shukla, C. Xiong, and S.-C. Zhu. A unified framework for human-robot knowledge transfer. In *2015 AAAI Fall Symposium Series*, 2015.

[17] P. Wei, Y. Zhao, N. Zheng, and S.-C. Zhu. Modeling 4d human-object interactions for event and object recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3272–3279, 2013.

[18] E. T. Mueller. *Commonsense reasoning: an event calculus based approach*. Morgan Kaufmann, 2014.

[19] S. C. Zhu and D. Mumford. Prior learning and gibbs reaction-diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(11):1236–1250, 1997.

[20] I. Caragiannis, S. Nath, A. D. Procaccia, and N. Shah. Subset selection via implicit utilitarian voting. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 151–157, 2016.

[21] K. Salkauskas. C1 splines for interpolation of rapidly varying data. *Rocky Mountain Journal of Mathematics*, 14(1), 1974.

[22] Y. Tassa, N. Mansard, and E. Todorov. Control-limited differential dynamic programming. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 1168–1175. IEEE, 2014.

[23] D. Xie, S. Todorovic, and S.-C. Zhu. Inferring dark matter and dark energy from videos. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2224–2231, 2013.

[24] S. Chernova and M. Veloso. Interactive policy learning through confidence-based autonomy. *Journal of Artificial Intelligence Research*, 34(1):1, 2009.

[25] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *Int. J. Comput. Vision*, 59(2):167–181, Sept. 2004. ISSN 0920-5691.

[26] C. Rother, V. Kolmogorov, and A. Blake. "grabcut": Interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.*, 23(3):309–314, Aug. 2004. ISSN 0730-0301.

[27] M.-K. Hu. Visual pattern recognition by moment invariants. *IRE transactions on information theory*, 8(2):179–187, 1962.

[28] A. Sen. Social choice theory. *Handbook of mathematical economics*, 3:1073–1181, 1986.

[29] X. Jiang, L.-H. Lim, Y. Yao, and Y. Ye. Statistical ranking and combinatorial hodge theory. *Mathematical Programming*, 127(1):203–244, 2011.

[30] N. Shukla. *Machine Learning with TensorFlow*. Manning Publications, 2017. ISBN 1617293873.