Learning Cost Functions For Motion Planning From Human Preferences

Artem Gritsenko[1] and Dmitry Berenson[1]

I. INTRODUCTION

Contemporary motion planners usually calculate the cost of the solution independently of the robot task. However, if we observe how humans perform motion, it seems that in many tasks some features of the trajectory are more important than others. For example, if you are manipulating a mattress in a house you may not pay attention to the distance to the obstacles that much due to the fact that the mattress is deformable, but you probably care about the execution time of the trajectory as the mattress is heavy. On the other hand, manipulation of some light and fragile object like a vase in the same house could consider the distance to obstacles more important than the execution time. Inspired by human behavior, we seek to improve the quality of motion planning by making the cost function task specific.

The problem of learning the cost function from human demonstration has been approached by Inverse Optimal Control (IOC) and Inverse Reinforcement Learning (IRL) techniques. The common idea in these methods is to learn the underlying Markov Decision Process [1] [2] [3] model of the observed motion. [4] uses linearly solvable MDPs and performs much faster than the previous algorithms by not solving the forward MDP optimization problem. The work in [5] is based on the Relative Entropy Policy Search and is able to learn good policies from a small number of demonstrations without any assumptions on an underlying dynamic model. Recent work in [6] tries to approach the problem in the local setting assuming the local optimality of demonstrated trajectories. In contrast to the previous work our approach does not solve MDP optimization problem, but uses the features of the trajectories and their ranking to produce the desired model. [7] and [8] learn the cost function by combining human preferences and Reinforcement Learning. [9] uses a human expert to provide a preference feedback without necessity to provide an optimal trajectory. [9] assumes the cost function is a linear combination of features, while we extend the cost function definition to any kind of function.

II. LEARNING A MODEL OF THE COST FUNCTION

A. Problem Statement

A common way to estimate the quality of a trajectory produced by a motion planning algorithm is to evaluate its cost using a function designed by the programmer. In our case we want to learn the cost function, so we need an alternative way to evaluate the robot motion when collecting training data. We propose to use human preferences to estimate the quality of a trajectory during training. As shown in previous work [10], humans are able to reliably choose between two alternative robot motions. So, we can provide a human with videos of two robot motions and allow him/her to pick the one that looks more appropriate. We can then aggregate the results of these pairwise comparisons to produce a ranking of the trajectories.

[1]Worcester Polytechnic Institute, Worcester, MA avgritsenko@wpi.edu, dberenson@cs.wpi.edu

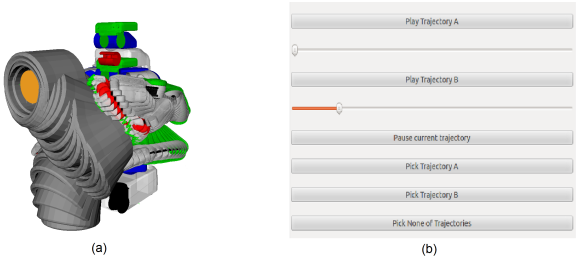

Fig. 1. (a) PR2 manipulating the wheel for unhang and put down actions in the OpenRave simulator; (b) GUI used for pairwise trajectory comparisons.

Let a set of trajectories for a given task be represented by a set of feature vectors $F = \{f_1, ..., f_m\}$. From pairwise comparisons, we obtain a ranking of the trajectories $R = \{r_1, ..., r_n\}$ where r_i is the rank of trajectory i. We wish to build a model of the cost function that explains the human-based ranking, or in a more mathematical definition, to find the mapping a:

$$a(F) = R. \tag{1}$$

B. Proposed Method

The first step is to generate the set of trajectories for a specific task. We used a task where robot needed to unhang a wheel from a car and put it down in front of it (Fig.1a). The features that are extracted from the generated trajectories are: 1) trajectory duration, 2) distance to the obstacles and 3) the length of the path that the wheel traveled.

To calculate the distance to the obstacles we used the Signed Distance Field in the workspace. The value of the distance feature in a certain robot configuration is the minimal distance to the obstacles along all the links of the robot we are interested in (usually links of the active manipulator). The value of the distance to obstacles feature for the trajectory is the sum of the values of the feature in each waypoint. The third feature captures the length of the path in terms of translation of the wheel. In many cases, this does not correlate with the trajectory duration (feature 1).

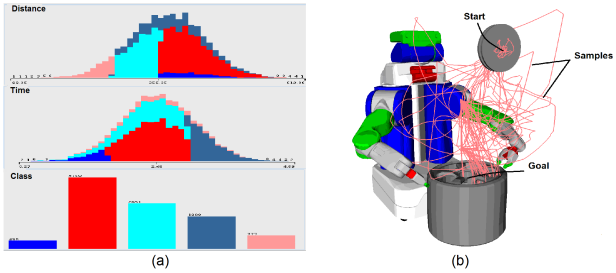

Fig. 2. (a) Distribution of the features according to the classes in the generated model. From bottom to top: number of instances for each of the classes; distribution of the data by time; distribution of data by distances, in color the number of instances assigned to each class is shown; (b) trajectories used for simulation experiments, which are generated for PR2 unhang and putdown wheel actions in OpenRave.

To produce the ranking we constructed a GUI that provides a human with two trajectories of the motion and allows him/her to pick between them (Fig.1b). To aggregate the pairwise comparisons into the total ranking of the trajectories the merge sort approach was used.

The next step is to find the mapping between features and ranking (Eq.1). We represent the trajectory ranking in two ways: as continuous values and as the class labels. In the first case we assign each trajectory a cost c in the range [0,1] such that 1 is the best rank and 0 is the worst. The second way of representing the ranking assigns a class label to each trajectory based on its rank. The problem here is that we need to define the number of classes and the threshold for every class. It could be done by assigning classes arbitrary or by making an additional run of demonstrations for the human judge that will allow choosing the thresholds for the classes. So far, we decide the number of classes and thresholds manually.

We propose to learn the model of the cost function using a decision tree-based approach. The main advantage of decision trees is that they implicitly perform feature selection by discarding irrelevant features. Decision trees can also capture nonlinear dependencies between variables, as opposed to, say, linear regression. For the class-based ranking we used conventional decision trees (C4.5 algorithm) [11]. For the continuous-cost ranking the model trees M5P algorithm [12] was used. This algorithm builds linear models in the leaves of the tree.

III. EXPERIMENTS

We tested the proposed method in two ways: 1) using data generated from a known distribution and 2) using data from a real planner and human comparisons.

A. Experiments with data from a known distribution

The first set of experiments was done to show that the approach works in principal. For this case we generated the set of trajectories' features (distance to obstacles and trajectory duration) from the normal distribution. The data were ranked with a simple cost function model that could reflect real world preferences (Fig. 2a). For example, if distance to the obstacles is very low (lower than 150mm) then the trajectory was assigned to the worst class and ranked descending by distance. On the other hand, if the distance was bigger than some threshold (for example 260mm) then distance is not very important and trajectories were ranked by time. We hypothesize that such splits can reflect real human preferences.

TABLE I
EXPERIMENTAL RESULTS

	Data from known distribution	Human-ranked data (training)	Human-ranked data (cross-validation)
Classification problem			
Accuracy	99.96%	90.62%	68.75%
Regression problem			
PCC	0.99	0.90	0.68
RAE	1.17%	41.7%	48.6%

TABLE II
CONFUSION MATRIX FOR HUMAN-RANKED DATA (CROSS-VALIDATION)

classified as				
a	b	c	d	
7	1	0	0	a
1	4	3	0	b
1	2	4	1	c
0	1	0	7	d

Experiments with 10,000 datapoints from the known distribution showed that decision and model trees can reproduce the underlying model of the cost function very reliably (Tab. I), which means that if our hypothesis about human preferences was true then our decision tree model can represent the cost function. For classification problem we use accuracy and confusion matrices as the measures of performance. For the continuous output the Pearson Correlation Coefficient (PCC) between the ground-truth and estimated score vectors is calculated as well as the Relative Absolute Error (RAE).

B. Experiments with human-ranked trajectories

To generate data for human comparisons the robot planned to unhang the wheel from the hub and put it down. This task contains complicated dual-arm constrained manipulation, which is non-trivial motion planning problem itself. 32 trajectories were generated with the CBiRRT planner [13] (Fig.2b). The following features of the trajectories were extracted: 1) distance to obstacles and 2) the length of the path that the wheel traveled (the trajectory duration feature was removed by correlation-based feature selection). We evaluated the decision trees learned from the ranked training data both on the training data and using leave-one-out cross-validation. The results are summarized in the Table I. Though for the classification problem the accuracy of cross-validation is not very high, we can notice from the confusion matrix (Tab. II) that misclassified instances are labeled with the class adjacent to the original. This could be caused by some noise in human pairwise comparisons or by the small size of the training data.

IV. CONCLUSION

We have presented an approach for learning a cost function for robot motion based on human preferences. Our method is based on the human ranking of the trajectories that is used as training data for decision trees. We have tested it on simulated trajectories for the PR2 robot performing tire manipulation. The results showed that the algorithm works in principal for 'perfect' data and can reproduce the underlying

model of the cost function. On the human-ranked data the accuracy was not as high, but still very promising.

REFERENCES

[1] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the twenty-first international conference on Machine learning*. ACM, 2004, p. 1.

[2] D. Ramachandran and E. Amir, "Bayesian inverse reinforcement learning," *Urbana*, vol. 51, p. 61801, 2007.

[3] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning." in *AAAI*, 2008, pp. 1433–1438.

[4] K. Dvijotham and E. Todorov, "Inverse optimal control with linearly-solvable mdps," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010, pp. 335–342.

[5] A. Boularias, J. Kober, and J. R. Peters, "Relative entropy inverse reinforcement learning," in *International Conference on Artificial Intelligence and Statistics*, 2011, pp. 182–189.

[6] M. Kalakrishnan, P. Pastor, L. Righetti, and S. Schaal, "Learning objective functions for manipulation," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 1331–1336.

[7] J. Fürnkranz, E. Hüllermeier, W. Cheng, and S.-H. Park, "Preference-based reinforcement learning: a formal framework and a policy iteration algorithm," *Machine Learning*, vol. 89, no. 1-2, pp. 123–156, 2012.

[8] R. Akrour, M. Schoenauer, and M. Sebag, "April: Active preference learning-based reinforcement learning," in *Machine Learning and Knowledge Discovery in Databases*. Springer, 2012, pp. 116–131.

[9] A. Jain, B. Wojcik, T. Joachims, and A. Saxena, "Learning trajectory preferences for manipulators via iterative improvement," in *Advances in Neural Information Processing Systems*, 2013, pp. 575–583.

[10] M. Zucker, J. A. Bagnell, C. G. Atkeson, and J. Kuffner, "An optimization approach to rough terrain locomotion," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 3589–3595.

[11] J. R. Quinlan, *C4. 5: programs for machine learning*. Morgan kaufmann, 1993, vol. 1.

[12] Y. Wang and I. H. Witten, "Inducing model trees for continuous classes," in *Proceedings of the Ninth European Conference on Machine Learning*, 1997, pp. 128–137.

[13] D. Berenson, S. S. Srinivasa, D. Ferguson, and J. J. Kuffner, "Manipulation planning on constraint manifolds," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, 2009, pp. 625–632.