Learning from Helicopter Trajectories
Part I: Coordinated Helicopter Turns

Nithin Vasisth* Vishal Dugar† Sanjiban Choudhury†
Sebastian Scherer†

August 28, 2016

CMU-RI-TR-16-xxx

Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

* Dept. of Mechanical Engg, Indian Institute of Technology, Bombay
† The Robotics Institute, Carnegie Mellon University

Abstract

The aim of this series of work is to develop a motion planning architecture for a fully autonomous helicopter using sensor data from existing flight demonstrations and to contribute to the advancement of autonomous capabilities for VTOL air systems.
In this report, we consider the problem of learning the pilot's preferences for coordinated helicopter turns using a motion planner developed in our previous work. Specifically, we aim to learn the optimal set of parameters for the motion planner such that the turns executed by our planner accurately represent the ones demonstrated by the pilot. We present a basic parameter tuning algorithm for the planner's cost function which gives satisfactory results for both self-generated and actual demonstrations of the coordinated turns.

1 Introduction

Traditional approaches to robot control derive policies based on the mathematical model of the domain dynamics. Developing such a model often requires considerable expertise and the derived policies heavily rely on the accuracy of the developed model. Therefore, any approximations to the model can significantly affect the optimality of the derived policies. This also implies that for any given application, a developer has to account for all the possible scenarios, however unlikely, and incorporate a relevant and robust control architecture to address the issue. The sensor noise, actuator limits, dynamic constraints, incomplete observation of the environment and in the case of non-deterministic policies, uncertainty about action add to the complexity of the problem. Therefore, the designing of a real-time motion planning system for an application is an open problem.

Our work focuses on developing a real-time motion planning architecture for a fully autonomous helicopter that has to fly smoothly in unknown environments. In an earlier work, we looked at a planner ensemble - a set of complementary planners that leverage a diverse set of assumptions, to address some of the problems mentioned above [10].

The approach, although giving a feasible real-time solution to the problem, sometimes outputs trajectories that are hard to execute. This problem can be circumvented by analyzing existing flight demonstrations to find the way a pilot approaches a given situation. We can either improve heuristics for our existing planning system or explore completely different paradigms for the same. In this report, we attempt to review relevant literature on one such paradigms: Learning from Demonstrations(LfD), with a focus on Inverse Reinforcement Learning(IRL), to establish the beginning of our research in that direction.

The problem of learning a task given example demonstration has been studied under various names, some of the common names being Learning from Demonstrations(LfD), Imitation Learning and Apprenticeship Learning. Based on the policy derivation techniques the work can be broadly classified as follows:

- Direct Approaches: Demonstration data is used to directly learn a policy, i.e a mapping from the robot's state observations to actions
- Indirect Approaches: Demonstration data is used learn an accurate representation of the task, which is used to derive an optimal policy or a sequence of actions; The representation could be a transition function, a reward function, a set of rules about the environment or a mixture of these

The direct approaches of learning a policy work well when the underlying system dynamics of the environment is unimportant and the agent has to simply mimic the expert trajectory. But these might not be useful in the cases like highway driving or navigating through a crowd, as the environment pattern

may be different each time. On the other hand, Indirect methods are difficult to represent and are usually resource-intensive.

Specific to our problem, helicopter trajectories vary widely between demonstrations and therefore simple applications of supervised learning approaches to mimic trajectories wouldn't be able to justify all the observations and the learned policy wouldn't be a true representation of the task. Therefore, we aim to recover something more intrinsic than a simple mapping from features to actions, like a reward function an agent is trying to optimize during the task.

In this paper, we restrict ourselves to the task of learning the cost function (or the negative reward function) for coordinated helicopter turns (Figure 3.1), that is compatible with a planner developed in our previous work and is able to represent the observed pilot demonstrations "well".

2 Related Work

There is a large body of work that attempts to learn a task given example behaviors, be it in the form of a reward function, policy or the value function. In this section, we present an overview of different paradigms used to approach this problem that are relevant to our work. For a comprehensive review of the methods, reader is directed to [4], [7] and [31].

Lot of the approaches in Learning from Demonstrations (LfD) try to directly mimic the expert through supervised learning algorithms. Regression techniques have been successfully applied to autonomous navigation [22], human-to-robot skill transfer [15],[5], aircraft simulations to mimic flying[16], etc. We now look at a body of work that has emerged in the past decade, involving the recovery of a reward function given expert demonstrations. This brand of techniques is based on the fact the reward function is a robust, concise and the most transferable way of defining a task.

2.1 Original IRL Algorithms

Inverse reinforcement learning can be informally characterized as follows [26]:

Given 1) the measurement of an agent's behavior over time, in a variety of circumstances, 2) if needed, measurements of the sensory inputs to that agents ; 3) if available, a model of the environment
Determine The reward function being optimized

One of the earliest studies related to IRL in the Machine Learning community was considered in setting of Markov decision processes [21]. The study characterizes the set of all reward functions for which the given agent behavior is optimal, and provides a set of algorithms for solving the problem.

A later work [3] builds upon the previous work with an application to apprenticeship learning. The algorithm returns a reservoir of policies that performs approximately as good as the expert's demonstrations. The authors were successful in developing a car driving simulation that learns different demonstrated driving styles.

Stanford helicopter project is one of the most notable applications in this field. A series of challenging aerobatic maneuvers were performed using the apprenticeship IRL theory [1]. Further, it was shown that by learning a desired trajectory from a number of sub-optimal demonstrations, helicopter performance can be greatly improved, even exceed expert performance [12], giving a good modeling technique for incomplete or noisy observations. IRL was also used in learning parking styles for an actual robotic car [2] and model pedestrian behavior in a robot application [11].

Inspite of these successes, there are some limitations with respect to the original IRL algorithms like reward function degeneracy due IRL being an ill-posed problem [21]; generalization errors due to noisy, incomplete or small number of observations that may be restricted to certain situations; sub-optimal policies due linear approximation of rewards; resultant policies being inherently stochastic (this can be a problem if the expert intention may have been deterministic); and also being computationally expense due to repeated solving of RL problems for every new reward function. [31]

2.2 Other Variants of IRL

However, there have been some variants of the original IRL algorithms that manage to tackle some of the deficiencies mentioned above. We briefly summarize some of the methods as mentioned in the [31]:

- Maximum margin planning[25]: This work learns the mapping from features to cost function in the structured large margin framework [29]. MMP also adds a degree of robustness with the use of a loss-augmented cost function. MMP is agnostic about the underlying MDP, returns a single policy instead of a mixture of policies and with the availability of a fast specialized algorithm to compute the minimum of the loss-augmented cost function, we get linear convergence rates when used in batch settings. This method was extended to learn non-linear reward functions and a LEArning and seaRCH (LEARCH) algorithm was introduced [24]

- Bayesian IRL models: This models the observed state sequence using a Bayesian framework, assuming that the expert is acting greedily according to the current reward function, resulting in a deterministic optimal policy (Ramachandran and Amir, 2007). The assumption of a linear reward function is removed by assigning a Gaussian prior and it has been shown

to be robust when dealing with noisy, incomplete and small number of observations [23]

- Maximum entropy models: Similar to Bayesian models, these models use a probabilistic approach. The optimal reward function, however, is obtained by maximizing the likelihood of observed trajectory through through maximum entropy [32]. By focusing on the distribution over the trajectories rather than action, these models avoids the problem of label bias [18] i.e avoids the case when the optimal policy with the highest reward may not have a high probability. The algorithm was extended to POMDP setting [20] and a model free setting [8]. This method was further extended to incorporate non-linear reward functions using deep neural networks [30]
- Active Learning Models: This deals with the cases when the demonstrations supplied are restricted or incomplete, and the agent queries the expert when the decision uncertainty rises to a certain level [13] [19]. Such methods can also help in reducing the number of demostrations required to learn the reward function

2.3 Other Relevant Studies

Methods have been developed to deal with sub-optimal expert demonstrations by learning the unknown desired trajectory from sub-optimal demonstrations [12], viewing the problem from a game-theoretic approach [27] and modeling the observations under a POMDP setting [20]. Hierarchical models have been implemented for the expert to instruct the agent more easily [17] and there are examples of agents performing well by deliberately avoiding the expert's mistakes [14]

Some other works consider the problem of extracting features from given demonstrations [9] and generating more expert demonstrations through parameterization of the given demonstrations [28]. Since, learning what features to represent is a critical part of the problem definition, one can also look into the deep learning literature [6].

3 Parameter Tuning for Helicopter Turns

In this section we consider the problem of learning a cost function for coordinated helicopter turns given flight demonstrations. We do so by estimating the parameters for the cost function of a motion planner that takes into account the dynamics of a coordinated turn, such that the trajectories executed by our planner resemble the demonstrated trajectories.

The flight demonstrations being used have been accumulated over years of trial runs and unit tests. These are performed by a trained pilot flying a full-scale helicopter, trying to follow the path generated by our planning system.

We have the following information available about the helicopter for the training phase: The position (x, y, z), orientation $(roll, pitch, yaw/heading)$, their derivatives and their double derivatives. We post process the given information to identify the kinematic state of the helicopter i.e whether it is turning, taking off, landing, etc., and we use this information to segregate an entire run into different components.

In the next few subsections, we describe the working of the planner and show that a simple parameter tuning algorithm is able to find a set of parameters that resemble the post processed data well. Let us first begin by describing the problem statement.

3.1 Problem Description

Given the flight demonstrations, our aim is to design either an absolute or a piece-wise reward function (or cost function) such that it represents the pilot's intentions for a given situation. This means that the reward function when used with a motion planner, generates trajectories that are "close" to the ones executed by the pilot in an identical scenario.

Since this report focuses on coordinated turns, let us formally define the term: A coordinated turn is maneuver performed by an air system to change the heading angle of the flight by altering the bank the angle of the system. The fact that the change in heading angle occurs only due to a simultaneous change in bank angle differentiates a coordinated turn from the non-coordinated turn. Figure 3.1 represents an ideal 90 degree coordinated turn for any VTOL system.

A coordinated turn obeys the following motion dynamics:

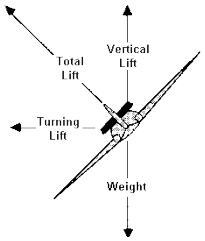
$$tan(\theta) = \frac{v}{g} * \dot{\psi} \tag{1}$$

where θ is the bank angle, v is the horizontal velocity w.r.t. ground, g is the acceleration due to gravity and $\dot{\psi}$ is the rate of change of heading angle

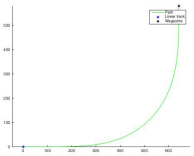
We now define a criteria to measure the performance of a resultant trajectory using the following conjecture:

Conjecture: For a fixed velocity, any trajectory following the motion dynamics assumption of an ideal coordinated turn (Equation 1; Figure 3.1) can be characterized by only its maximum roll rate (or maximum bank angle) and the change in its heading angle.

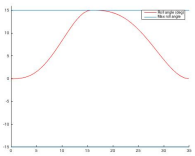
Let $X \subset \mathbb{R}^n$ be the configuration space of the helicopter and the trajectory $\xi : [0, 1] \rightarrow X$ be a smooth mapping from time to configuration.



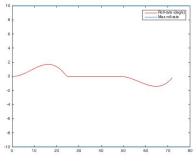
(a) Coordinated turn



(b) Ideal y v/s x path for 90 degree turn



(c) Ideal bank angle v/s time curve



(d) Ideal roll rate v/s time curve

Figure 3.1: Coordinated Turn

Definition 3.1. $\Upsilon(\xi) \triangleq \max_{0\leq t\leq 1} \dot{\phi}(t) - \min_{0\leq t\leq 1} \dot{\phi}(t)$
where $\dot{\phi}(t) \in X(t)$ *is the roll rate for the trajectory* ξ *at time* t

Definition 3.2. $\Delta\Psi(\xi) \triangleq \psi(1) - \psi(0)$
where $\psi(t)$ *is the heading angle for the trajectory* ξ *at time* t

Criteria: A generated turn ξ_{gen} having $\Delta\Psi(\xi) = \text{k}$, is acceptable if the following condition is satisfied:

If Ξ is the set of demonstrated turns such that $\Delta\Psi(\xi) = \text{k}$; $\forall \xi \in \Xi$
Then, $|\Upsilon(\xi_{gen}) - \text{mean}_{\xi\in\Xi}(\Upsilon(\xi))| \leq \epsilon$ where, ϵ is arbitrarily small

In practice, we can define ϵ heuristically or in terms of $\text{variance}_{\xi\in\Xi}(\Upsilon(\xi))$ and any demonstration ξ is in Ξ if $\Delta\Psi(\xi)$ is approximately equal to k.

In other words, considering trajectories having a fixed heading change, we take the difference between the maximum and the minimum roll rates in the roll rate profiles as the characteristic of a trajectory. And the objective of our parameter tuning algorithm is to minimize the difference in this quantity between the generated and the observed trajectories.

3.2 Planner Description

Our motion planner poses the problem as a constrained non-linear optimization problem. The cost function or the negative reward function depends on the length and the overall smoothness of the trajectory (defined here as the summation of roll rates maneuvered throughout the path) and the parameters of the cost function are the weights assigned to these.

$$FinalCost = w_t * total\ length + w_s * \sum roll\ rates \tag{2}$$

The planner obeys the dynamics of a coordinated turn as described in Equation 1, and also takes into account the dynamical constraints like maximum permitted airspeed, bank angle, horizontal acceleration along with their higher order derivatives.

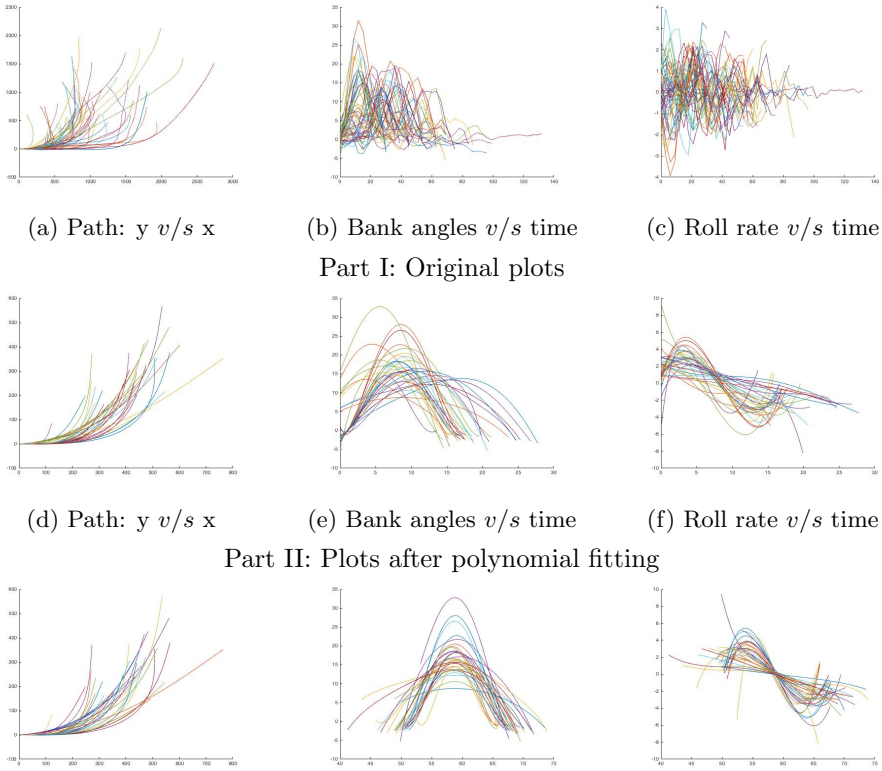
The planner solves this optimization problem by calling a constraint nonlinear optimizer. In our case, we use the function *fmincon* in MATLAB.

3.3 Post Processing helicopter turns

As we see in (Figure 3.2, Part I), the turns vary widely between demonstrations. Although individually each trajectory fleetingly resembles the ideal trajectory assumptions as described in figure 3.1, when plotted together, they do not exhibit any distinctive behaviour. Therefore, we post process the demonstrations by smoothening and time-space alignment to check if we get some pattern from the data that we can learn. This subsection covers this process in detail.

- Step 1: Space Alignment
  1. Extract all the turns close to 90 degrees
  2. Align the turns so that they start from the same point with the initial heading angle being zero.
- Step 2: Profile Smoothing by polynomial fitting. (See Figure3.2, Part II)
  1. Fit a polynomial for the bank angle and the velocity profile. In our case, we use an eleventh degree polynomial.
  2. Discard examples having more than one peak in the smoothed bank angle profile
  3.* Crop the trajectories at the inflection points surrounding the peak
  4.* Smooth the profile such that bank profile the starts from zero and decays back to zero gradually
  5.* Recalculate the entire trajectory using the new bank angle and velocity profile using motion dynamics assumptions of a coordinated turn (Equation 1).

  *Steps 3-5 are irrelevant to our parameter tuning algorithm, as this doesn't affect the maximum and the minimum roll rates. These steps are performed only for the final visual comparison. Actual helicopter demonstrations are never fully isolated, they are accompanied by residual values of other components of the flight path. These steps only ensure the removal of these residual affects.
- Step 3: Time alignment. (See Figure3.2, Part III)



(a) Path: y v/s x (b) Bank angles v/s time (c) Roll rate v/s time

Part I: Original plots

(d) Path: y v/s x (e) Bank angles v/s time (f) Roll rate v/s time

Part II: Plots after polynomial fitting

(g) Path: y v/s x (h) Bank angles v/s time (i) Roll rate v/s time

Part III: Plots after time alignment

Figure 3.2: Post Processing of Coordinated Turns

1. Align the bank angle profiles such that the global maximas for each trajectory occur at the same time
2. We discard any outliers that do not fit our assumed bank angle or roll rate profile (See figure 3.1)

3.4 Basic Parameter Tuning Algorithm

Algorithm 1 describes the functioning of our the algorithm. As we discussed earlier (Section 3.1), for a fixed heading change, we take the difference between the maximum and the minimum roll rates in the roll rate profiles as the characteristic of a trajectory($\Upsilon(\xi)$, as per Definition 3.1). The main crux of our method is that at each iteration we try to minimize the difference in this quantity between the generated and the observed trajectories.

The advantage of our parameter definition is that they are intuitive to update given the current performance of an iteration. We are essentially looking at a pilot's trade off between length of the turn and smoothness of the turn, which are in inverse relation to each other. So, if Υ for our generated trajectory is less than that of demonstrated trajectories, we simply give more preference to shortening the length of the trajectory and less preference to the overall smoothness of the turn. Therefore, our parameter update at each iteration is a simple function of Υ

Algorithm 1 Parameter Tuning algorithm for our motion planner

Parameters:
    ws: Smoothness weight of the curve
    wt: Time weight of the curve
    scale: Scale factor for parameter update
Input:
    ξ_0 : Post-processed demonstrated trajectory set
procedure PARAMETER TUNING
    $g \leftarrow \infty$
    while $g < \delta$ **do**
        $g_{\text{prev}} \leftarrow g$
        ξ = Trajectory Planner (wt,ws)
        g = Difference in roll rate profile (ξ,ξ_0)
        ws $\leftarrow$ ws*exp(tanh(-g/scale))
        wt $\leftarrow$ wt*exp(tanh(g/scale))
        ws,wt $\leftarrow$ NORMALIZE (ws,wt)
        if $g * g_{\text{prev}} < 0$ **then**
            scale = scale * 2
        end if
    end while
end procedure

3.5 Result on self-generated turns

To demonstrated the algorithms ability to converge and learn parameter for different settings, we constructed three trajectory sets: One set placing a relatively large weight on the length of the trajectory, one placing a relatively large weight on the smoothness of the trajectory and another in between the two.

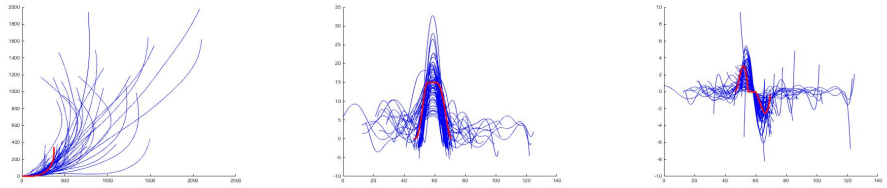
We can see that the algorithm is able to identify the initial set of parameters with sufficient accuracy. See Table 1

3.6 Result on post-processed demonstrations

We see that our algorithm manages to fit the post-processed demonstrations well. Figure 3.3. Unlike, the self generated turns, we do not have any ground

	Set 1		Set 2		Set 3	
	wt	ws	wt	ws	wt	ws
Traj 1	0.0886	999.9114	10.6407	989.3592	999.3089	0.6910
Traj 2	0.0928	999.9071	10.9920	989.0079	999.4328	0.5671
Traj 3	0.0847	999.9152	10.2267	989.7732	999.4407	0.5592
Traj 4	0.1103	999.8896	10.7380	989.2619	999.6965	0.3034
Traj 5	0.0953	999.9046	10.5624	989.4375	999.7513	0.2486
Traj 6	0.1131	999.8868	10.4123	989.5876	999.5622	0.4377
Traj 7	0.1028	999.8971	10.3371	989.6628	999.5690	0.4309
Traj 8	0.0864	999.9135	10.6074	989.3925	999.9741	0.0258
Traj 9	0.0921	999.9078	10.4064	989.5935	999.6770	0.3229
Traj 10	0.0945	999.9054	10.3274	989.6725	999.0806	0.9193
Learned	0.0887	999.9113	9.8783	990.1217	929.7884	70.2116
Percentage Error*	1.33 x 10^{-4}		0.0026		1.56 x 10^{-4}	

Table 1: Learning parameters on different sets of self-generated data
* "Percentage Error" is defined in terms of the roll rate profile difference



(a) Path: y v/s x (b) Bank angles v/s time (c) Roll rate v/s time

Figure 3.3: Blue: Post-processed demonstrations; Red: Final Trajectory obtained using the learned parameters

truth to compare the validity of our results. We can always use the classic machine learning technique of dividing our demonstrations into training and test set to validate our algorithm.

3.7 Conclusion

The main issue we faced was a lack of a good evaluation metric for actual trajectory demonstrations. Simply considering the mean of the maximum roll rate of all demonstrations may not be representative of the entire set. And this method of parameter update is very specific to the planner and the problem of helicopter turns.

The learned parameter setting is good enough if the resultant trajectory respects the dynamics of a coordinated turn and is close to expert demonstration. The dynamics being inherent in our planning algorithm is always satisfied, and

we notice that the resulting trajectory fits the data well, in both self-generated and actual turns. Although in case of self-generated trajectories, we have a good evaluation criteria, and we notice low training and test error rates.

Here, we specifically consider only the right-angled turns. We also discard the turns that do not exhibit the particular roll and roll-rate profile characteristic of a perfect coordinated turn. That is, we only consider cases when the pilot is making the entire turn at once and not in segments. We might need another representation of the problem to have all the data into consideration and generalize well.

4 Future Work

The aim of our work is to be able to mathematically represent the complete model trajectory of the helicopter, either piece-wise or complete, driven by demonstrated data and robust to a variety of situations. This representation would likely be in terms of a cost map or a reward function that complements our existing ensemble of planners.

In this work, we have dealt with the simple case of helicopter turns, specific to an angle. But this method can be generalized to all helicopter turns, by treating the cases in a piece-wise manner. In the future, we will attempt to solve the problem for other segments of the helicopter trajectory and utilize a motion planner with a more flexible behavior.

The main challenge going ahead will be that of extracting features given a set of demonstrations and quantifying what does it mean to be optimal, i.e coming up with a good performance measure with respect to the given task of flying a helicopter.

References

[1] Pieter Abbeel, Adam Coates, Morgan Quigley, and Andrew Y. Ng. An application of reinforcement learning to aerobatic helicopter flight. In *In Advances in Neural Information Processing Systems 19*, page 2007. MIT Press, 2007.

[2] Pieter Abbeel, Dmitri Dolgov, Andrew Ng, and Sebastian Thrun. Apprenticeship learning for motion planning, with application to parking lot navigation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-08)*, Nice, France, September 2008. IEEE.

[3] Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *In Proceedings of the Twenty-first International Conference on Machine Learning*. ACM Press, 2004.

[4] Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robot. Auton. Syst.*, 57(5):469–483, May 2009.

[5] Christopher G. Atkeson and Stefan Schaal. Robot learning from demonstration. In *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, pages 12–20, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[6] Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. Unsupervised feature learning and deep learning: A review and new perspectives. *CoRR*, abs/1206.5538, 2012.

[7] A. Billard and D. Grollman. Robot learning by demonstration. 8(12):3824, 2013. revision:138061.

[8] A. Boularias, J. Kober, and J. Peters. Relative entropy inverse reinforcement learning. In *JMLR Workshop and Conference Proceedings Volume 15: AISTATS 2011*, pages 182–189, Cambridge, MA, USA, April 2011. MIT Press.

[9] Shenyi Chen, Hui Qian, Jia Fan, Zhuo-Jun Jin, and Miaoliang Zhu. Modified reward function on abstract features in inverse reinforcement learning. *Journal of Zhejiang University - Science C*, 11(9):718–723, 2010.

[10] Sanjiban Choudhury, Sankalp Arora, and Sebastian Scherer. The planner ensemble: Motion planning by executing diverse algorithms. In *IEEE International Conference on Robotics and Automation, ICRA 2015, Seattle, WA, USA, 26-30 May, 2015*, pages 2389–2395, 2015.

[11] Shu Yun Chung and Han-Pang Huang. A mobile robot that understands pedestrian spatial behaviors. In *IROS*, pages 5861–5866. IEEE, 2010.

[12] Adam Coates, Pieter Abbeel, and Andrew Y. Ng. Learning for control from multiple demonstrations. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 144–151, New York, NY, USA, 2008. ACM.

[13] Robert Cohn, Michael Maxim, Edmund H. Durfee, and Satinder P. Singh. Selecting operator queries using expected myopic gain. In Jimmy Xiangji Huang, Ali A. Ghorbani, Mohand-Said Hacid, and Takahira Yamaguchi, editors, *IAT*, pages 40–47. IEEE Computer Society Press, 2010. 978-0-7695-4191-4.

[14] Daniel H. Grollman and Aude Billard. Donut as i do: Learning from failed demonstrations. In *ICRA*, pages 3804–3809. IEEE, 2011.

[15] Gregory Z. Grudic and Peter D. Lawrence. Human-to-robot skill transfer using the spore approximation. In *In Proc. IEEE Int. Conf. Robot. Autom. (ICRA*, pages 2962–2967, 1996.

[16] Andrew Isaac and Claude Sammut. Goal-directed Learning to Fly. In *Proceedings of the 20th International Conference on Machine Learning*, pages 258–265. AAAI Press, 2003.

[17] J. Zico Kolter, Pieter Abbeel, and Andrew Y. Ng. Hierarchical apprenticeship learning with application to quadruped locomotion. In John C. Platt, Daphne Koller, Yoram Singer, and Sam T. Roweis, editors, *NIPS*, pages 769–776. Curran Associates, Inc., 2007.

[18] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[19] Manuel Lopes, Francisco S. Melo, and Luis Montesano. Active learning for reward estimation in inverse reinforcement learning. In Wray L. Buntine, Marko Grobelnik, Dunja Mladenic, and John Shawe-Taylor, editors, *ECML/PKDD (2)*, volume 5782 of *Lecture Notes in Computer Science*, pages 31–46. Springer, 2009.

[20] Francisco S. Melo and Manuel Lopes. *Learning from Demonstration Using MDP Induced Metrics*, pages 385–401. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[21] Andrew Y. Ng and Stuart J. Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 663–670, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[22] Dean A. Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural Computation*, 3:97, 1991.

[23] Qifeng Qiao and Peter A. Beling. Inverse reinforcement learning with gaussian process. *CoRR*, abs/1208.2112, 2012.

[24] Nathan Ratliff, David Silver, and J. Andrew (Drew) Bagnell. Learning to search: Functional gradient techniques for imitation learning. *Autonomous Robots*, 27(1):25–53, July 2009.

[25] Nathan D. Ratliff, J. Andrew Bagnell, and Martin A. Zinkevich. Maximum margin planning. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 729–736, New York, NY, USA, 2006. ACM.

[26] Stuart Russell. Learning agents for uncertain environments (extended abstract). In *Proceedings of the 11th Annual Conference on Computational Learning Theory (COLT)*, pages 101–103, 1998.

[27] Umar Syed and Robert E. Schapire. A game-theoretic approach to apprenticeship learning. In John C. Platt, Daphne Koller, Yoram Singer, and Sam T. Roweis, editors, *NIPS*, pages 1449–1456. Curran Associates, Inc., 2007.

[28] Jie Tang, Arjun Singh, Nimbus Goehausen, and Pieter Abbeel. Parameterized maneuver learning for autonomous helicopter flight. In *ICRA*, pages 1142–1148. IEEE, 2010.

[29] Ben Taskar, Vassil Chatalbashev, Daphne Koller, and Carlos Guestrin. Learning structured prediction models: A large margin approach. In *Proceedings of the 22Nd International Conference on Machine Learning*, ICML '05, pages 896–903, New York, NY, USA, 2005. ACM.

[30] Markus Wulfmeier, Peter Ondruska, and Ingmar Posner. Deep inverse reinforcement learning. *CoRR*, abs/1507.04888, 2015.

[31] Shao Zhifei and Er Meng Joo. A survey of inverse reinforcement learning techniques. *International Journal of Intelligent Computing and Cybernetics*, 5(3):293–311, 2012.

[32] Brian D. Ziebart, Andrew Maas, J. Andrew (Drew) Bagnell, and Anind Dey. Maximum entropy inverse reinforcement learning. In *Proceeding of AAAI 2008*, July 2008.