Nonparametric Bayesian Reward Segmentation for Skill Discovery Using Inverse Reinforcement Learning

Pravesh Ranchod[1], Benjamin Rosman[1,3], George Konidaris[2]

***Abstract*— We present a method for segmenting a set of unstructured demonstration trajectories to discover reusable skills using inverse reinforcement learning (IRL). Each skill is characterised by a latent reward function which the demonstrator is assumed to be optimizing. The skill boundaries and the number of skills making up each demonstration are unknown. We use a Bayesian nonparametric approach to propose skill segmentations and maximum entropy inverse reinforcement learning to infer reward functions from the segments. This method produces a set of Markov Decision Processes (MDPs) that best describe the input trajectories. We evaluate this approach in a car driving domain and a simulated quadcopter obstacle course, showing that it is able to recover demonstrated skills more effectively than existing methods.**

I. INTRODUCTION

Programming agents in complex domains such as computer games and robotics can be prohibitively time consuming, requiring extensive engineering expertise. Learning from Demonstration (LfD) is an alternative approach which can reduce this complexity by allowing a human expert to demonstrate tasks to an agent. The approach has been used to teach agents to walk [1], perform helicopter manoeuvers [2] and play real-time strategy games [3].

LfD methods fall into two broad categories based on the approach taken to mimic the demonstrator. Methods either learn a policy which maps actions to states in a way that is consistent with the demonstrator, or attempt to learn a task description capturing the goal of the demonstrator [4]. For the latter category, it is common to learn an objective or reward function, which is the "most succinct, robust, and transferable definition of the task" [5]. A significant advantage of this approach, known as inverse reinforcement learning (IRL), is that a reward function is robust to changes in the environment and agent configuration, making it suitable for representing transferrable skills [6].

LfD algorithms typically search for a policy or reward that explains the entirety of a demonstration, making the assumption that demonstrators are performing a single monolithic task. This is problematic for two reasons. Firstly, it imposes the unnatural constraint that the demonstrator must repetitively perform a single task with a defined start and end. Secondly, the discovery of complex global reward functions is suboptimal in the context of transfer, as a complex task may be composed of skills that can be reused in other tasks.

Recently, attempts have been made to allow learning multiple skills from natural, unstructured demonstrations and have focused on the characterisation of skills as attempts to reach particular subgoal states [7], [4], [8]. This approach has shown good results, but suffers from an inability to represent the complex reward structures which may be encountered in practice.

We propose nonparametric Bayesian reward segmentation (NPBRS), a method to discover latent skills from unstructured demonstration trajectories. It works by segmenting the trajectories by identifying the skill employed in each segment through the reward function it is optimizing. We accomplish the segmentation by discovering the set of reward functions which results in the highest likelihood of the demonstration trajectories. Because we do not know how many skills are employed in a set of demonstration trajectories, we use a Bayesian nonparametric approach to segmentation, allowing the model complexity to be determined in a data driven manner. We also do not know that the trajectories have the same skill transition dynamics, as different contexts may use different sequences of common skills. We use the Beta Process Hidden Markov Model (BP-HMM) [9] as it allows for this complication.

NPBRS is able to identify multiple skills in unstructured demonstrations, identifying skill boundaries and recovering the reward functions associated with them. We show that it extracts suitable skills given demonstration trajectories in two domains, the car driving domain and the quadcopter gate domain.

II. BACKGROUND

A. Reinforcement Learning

Reinforcement learning problems are usually modelled as a Markov Decision Process (MDP) [5]. This is a control process defined as a tuple $M = (S, A, T, R)$, where S is the set of possible states of the environment, A is the set of actions available to an agent, $T(s, a, s')$ is the probability that taking action a from state s will result in a transition to state s'; and $R(s, a, s')$ is the reward obtained by the agent for taking action a in state s, causing a transition to state s'. Note that because T is stochastic, an action taken by an agent in a particular state need not transition to the same resulting state each time. The goal is to discover a policy

[1]P. Ranchod and B. Rosman are with the School of Computer Science and Applied Mathematics, University of the Witwatersrand, Johannesburg, South Africa.

[2]G. Konidaris is with the Departments of Computer Science and Electrical and Computer Engineering, Duke University, NC, USA.

[3]B. Rosman is also with the Council for Scientific and Industrial Research, Pretoria, South Africa.

This work is based on research supported in part by the National Research Foundation of South Africa for the grant 84392. Any opinion, finding and conclusion or recommendation expressed in this material is that of the authors and the NRF does not accept any liability in this regard.

$\pi = S \times A$ that maximizes the agent's cumulative expected reward.

B. Hierarchical Reinforcement Learning

Hierarchical reinforcement learning focuses on decomposing a task into smaller subtasks. The options framework [10] is a common approach to this. Options are temporally extended actions which can be used to represent skills.

Formally, an option o is a tuple (I_o, π_o, β_o), where $I_o \subseteq S$ is the initiation set of the option, which defines the states in which the option can be initiated, $\pi_o : S \times A \rightarrow [0,1]$ is the option policy which governs action selection during the execution of the policy, and $\beta_o : S \rightarrow [0,1]$ is the termination condition, which gives the probability of terminating the execution of the option when in state s.

Automated option creation methods must learn the policy corresponding to the option along with the initiation and termination sets. Methods exist for finding I_o and β_o given a segmentation [7]. We focus on discovering the option reward function R_o, and then treat the policy learning as a standard reinforcement learning task.

C. Inverse Reinforcement Learning

Inverse reinforcement learning (IRL) [11] is the process of determining a reward function from expert demonstration. IRL algorithms take as input an MDP\R which is a tuple (S, A, T) and a set of trajectories D where each trajectory $D^{(i)} = \{(s_1^{(i)}, a_1^{(i)}), (s_2^{(i)}, a_2^{(i)}), ...\}$ represents a sequence of state and action pairs produced by the expert. The goal is to find a reward function R such that D is likely to have been produced by an agent solving the MDP (S, A, T, R).

It is worth noting here that the log likelihood of D given policy π optimizing the reward function R can be written as $\log P(D|R) = \Sigma_i \Sigma_t \log \pi(a_t^{(i)}|s_t^{(i)})$. This makes it clear that the likelihood is not affected by the stochastic nature of T as the goal is to mimic the behaviour of the expert rather than the reaction of the environment.

A number of IRL algorithms have achieved good results [11], [12], [13]. We use maximum entropy inverse reinforcement learning [14] as tests indicated faster convergence than competing methods, but our method is easily extensible to other algorithms.

D. Subgoal State Methods

The Bayesian nonparametric inverse reinforcement learning algorithm is capable of learning multiple reward functions from unstructured demonstrations [20]. The method has been extended with a number of optimizations making it suitable for large state spaces [21], but is restricted to reward functions representing the goal of reaching particular subgoal states. These reward functions return a positive reward in a single state, and zero elsewhere.

This restriction is also present in the CST algorithm [7], which segments trajectories using changepoint detection with each segment mapping to a single value function. The method produces options which can be chained together with the termination set of an option corresponding to the initiation set of its successors. The discovered options have the goal of reaching these common subgoal states.

E. Switching Linear Dynamical Systems

Algorithms which model expert trajectories as a series of switches between linear dynamical systems have achieved good results in the context of human motion segmentation [15], [16], [17]. Hidden Markov Models (HMMs), which are Markov processes with unobservable hidden states known as modes, have been successfully used to model this switching behaviour [18]. In these, each mode represents a linear dynamical system. Methods exist which can discover the most likely sequence of modes, but commonly require that either the number of modes be prespecified (which is not realistic for unstructured demonstrations) or do not allow for modes to be repeated or shared across trajectories.

The Beta Process Hidden Markov Model (BP-HMM) [9] solves this problem by placing a beta process prior on the mode sequence. This allows for potentially infinitely many modes, with each time series exhibiting a subset of the total modes. Each time series can then switch between the modes it exhibits. This representation also encourages the sharing of modes between trajectories. The appropriate number of modes can be inferred from the data instead of being prespecified. In the Beta Process Autoregressive Hidden Markov Model (BP-AR-HMM), emissions are produced from a mode specific vector autoregressive (VAR) process.

The generative model for the BP-AR-HMM is

$$
\begin{aligned}
B|B_0 &\sim \mathrm{BP}(1, \beta_0) \\
X_i|B &\sim \mathrm{BeP}(B) \\
\pi_j^{(i)}|f_i, \gamma, \kappa &\sim \mathrm{Dir}([\gamma, ..., \gamma+\kappa, \gamma, ...] \otimes f_i) \\
z_t^{(i)} &\sim \pi^{(i)}_{z_{t-1}^{(i)}} \\
\mathrm{y}_t^{(i)} &= \sum_{j=1}^{r} A_{j, z_t^{(i)}} \mathrm{y}_{t-j}^{(i)} + \mathrm{e}_t^{(i)}(z_t^{(i)}).
\end{aligned}
$$

B, drawn from a beta process, provides a set of coin-flipping probabilities ω_k, with one global probability for each skill. These ω_k are then used to produce a Bernoulli process realization X_i for each trajectory i. The X_i distributions are used to construct a binary feature vector f_i that indicates which skills are present in the ith trajectory. The beta process formulation encourages shared skills among the trajectories, while also leaving room for any trajectory to leave out particular skills. The graphical model representation is shown in Figure 1.

Next, the feature constrained transition probability vector $\pi_j^{(i)}$ is drawn from a Dirichlet distribution, and defines the transition probabilities for time series i in mode j. The mode sequence is drawn at each time step t from the transition distribution of the mode at time step $t-1$. The observations are computed from the mode dependent linear dynamical system.

Inference is accomplished using a Markov Chain Monte Carlo (MCMC) algorithm which solves for the mode sequence $z_t^{(i)}$ [19]. The sampler alternates between: sampling

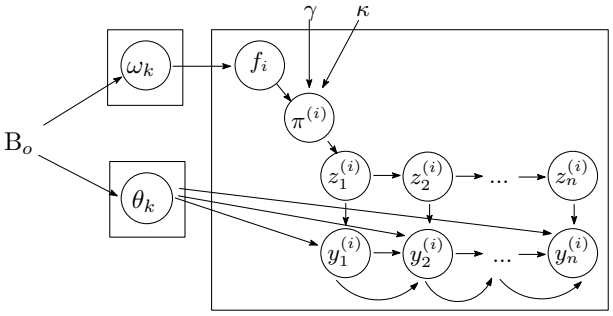

Fig. 1. Graphical model of the BP-AR-HMM.

the feature vector and mode sequence given the data and mode specific HMM emission parameters; sampling the HMM emission parameters from a matrix normal inverse-Wishart distribution given the data and mode sequence; and proposing birth/death moves which introduce new features or remove existing ones from the global feature set.

III. NONPARAMETRIC BAYESIAN REWARD SEGMENTATION

Our task is to identify repeated skills given a set of demonstration trajectories. Each trajectory may contain multiple skills, and common skills occur in multiple trajectories. We model these skills as reward functions. This is a departure from the use of subgoal states in previous work, and has the advantage of being more robust to changes in the environment and in the agent's configuration, which are important properties in the context of transfer. The use of reward functions rather than states also allows us to capture more complex subgoals that cannot be represented as a single target state, as illustrated in Figure 2. We introduce

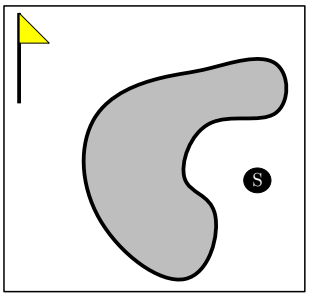

Fig. 2. If the goal of the agent is to get from position S to the flag, we could represent the subgoal as a target state. However, a single target state cannot capture the situation where an agent must also avoid the swamp in the middle.

nonparametric Bayesian reward segmentation (NPBRS), a method for segmenting trajectories from an MDP according to latent reward functions. The method combines the BP-HMM and IRL. Note that, like most IRL methods, it assumes knowledge of the transition function T.

The system is required to segment the demonstration trajectories into skills, with the important property that skills can be shared between trajectories, and that the number of skills is potentially infinite. We use the beta process and conjugate Bernoulli process in the same manner as the BP-HMM model, and draw features and mode sequences representing skills in the same way. However, we model the emissions as an MDP rather than as a VAR process.

Given the mode sequence $z_t^{(i)}$, we model the dynamics of the system as follows:

$$
\begin{aligned}
P(a)|z_t^{(i)} &= \frac{e^{\tau Q^{z_t^{(i)}}(y_{t-1}^{(i)},a)}}{\sum_a e^{\tau Q^{z_t^{(i)}}(y_{t-1}^{(i)},a)}} \\
a_t^{(i)} &\sim P(a)|z_t^{(i)} \\
y_t &\sim T(y_{t-1}, a_t^{(i)}).
\end{aligned}
\tag{1}
$$

We thus replace the step wherein HMM emission parameters A and e are sampled. Instead, we use IRL to determine the agent's reward function and accompanying policy, and calculate the transition probabilities from a combination of the policy and the known dynamics of the environment. The graphical representation is shown in Figure 3.

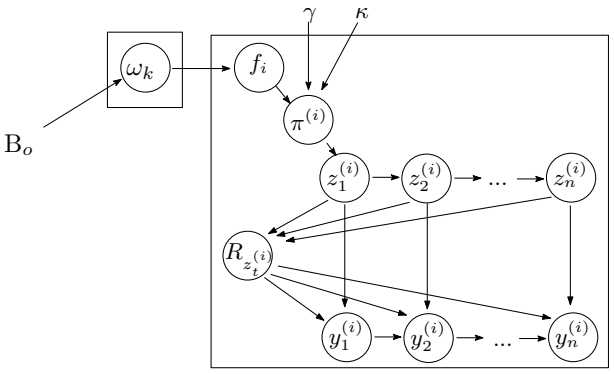

Fig. 3. Graphical model of the BP-HMM applied to MDPs. $R_{z_t^{(i)}}$ represents the reward function of the MDP in mode $z_t^{(i)}$.

We make use of the birth and death reversible jump Markov Chain Monte Carlo sampler developed for BP-AR-HMM [18]. The algorithm proposes skill birth and death moves and accepts or rejects them on the basis of the relative model likelihood. Skills can also be created using a split-merge procedure which introduces new skills by either combining or splitting existing ones [19]. The skill sequence is then block sampled using a variant of the forward-backward algorithm [22].

Our algorithm then models the skill specific dynamics by learning the action-value function associated with each skill ($Q^{z_t^{(i)}}$ in equation 1). We do this by grouping all subtrajectories by the skill assigned to them. We perform maximum entropy IRL on each skill, taking all of its subtrajectories as input paths. This process yields a reward function for each skill. We then use value iteration to determine $Q^{z_t^{(i)}}$, allowing us to determine the likelihood of the demonstration trajectories. This approach replaces the HMM emission model with an MDP, allowing us to use the segments to infer a policy to be used for control, rather than to recognise a change in

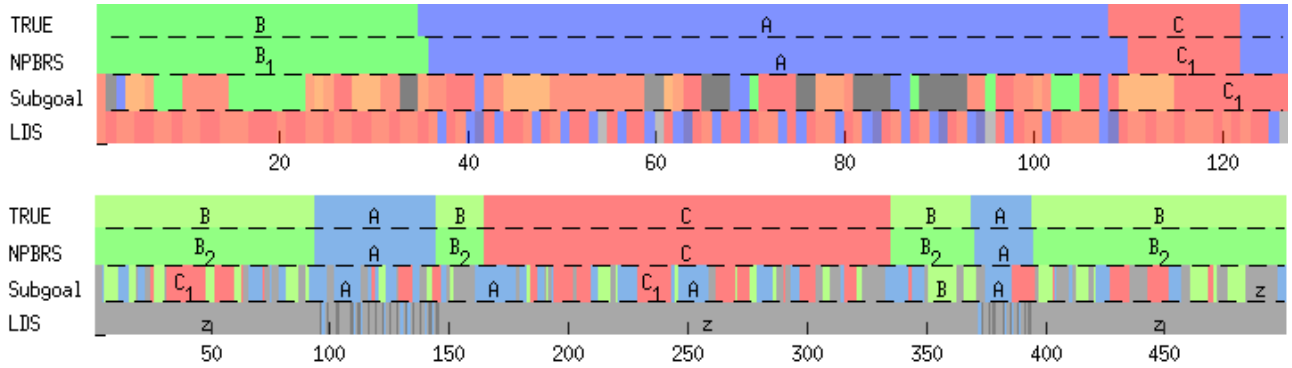

Fig. 4. Segmentation for the car driving domain for trajectory lengths 128 (top) and 500 (bottom), showing using segmentation based on NPBRS, subgoal states, and linear dynamical systems.

dynamics. It does, however, incur a large cost as we must perform IRL multiple times at each iteration of the sampler.

IV. EXPERIMENTS

A. Trajectory Generation

To test our method, we create a number of trajectories and attempt to segment them on the basis of the reward functions inherent at each timestep. To generate input, we create agents which employ particular reward functions.

For each of the two domains, we designed multiple reward functions representing skills. We then used each reward function to train an optimal agent using Q-Learning [23]. In order to generate trajectories containing a mixture of skills, we switched between the action selections of the trained agents. This selection is biased to be "sticky", as a skill is likely to be employed for multiple sequential timesteps. We generated 16 trajectories for each domain, and recorded the true switching sequence for later comparison.

We used these trajectories as input to the segmentation methods. We ran the sampler 10 times for 40 minutes each, and selected the segmentation with the highest log likelihood.

We evaluate the effectiveness of segmentation based on extracted rewards by comparing against methods based on subgoal states and linear dynamical systems. In order to remove the effect of other design decisions, we keep the probabilistic model and inference procedure the same for all the methods, and vary the segmentation likelihood calculation to obtain algorithms based on reward functions, subgoal states and linear dynamical systems.

B. Driving Domain

For our first experiment, we use a modified version of the car driving domain, commonly used as an IRL benchmark problem [11]. In this domain, the agent is driving on a three lane highway as shown in the screenshot in Figure 5, with each containing cars moving at a different speed.

The car controlled by the agent is moving faster than all other cars on the road and has three available actions: it can change lanes to the left, to the right, or stay in the current lane. The car is also able to drive on the verge to either side of the highway, in which there are no cars. Cars appear randomly in the distance, so it is possible for the highway to be completely blocked.

The state is described by 4 variables: the current lane, which includes the verge on either side, and 3 variables representing the distance to the closest car in each lane. If this distance is zero for the current lane, a collision has occurred. This distance was discretized to 6 distance values to simplify the learning process.

We designed three reward functions to encode different objectives:

- A: Hit as many cars as possible. The agent receives a reward of 500 for hitting a car, 0 otherwise.
- B: Drive in the right lane, but change lanes to avoid collisions. The agent receives a reward of 10 for being in the right lane, -500 for hitting a car and 0 otherwise.
- C: Drive in the left lane, but change lanes to avoid collisions. The agent receives a reward of 10 for being in the left lane, -500 for hitting a car and 0 otherwise.

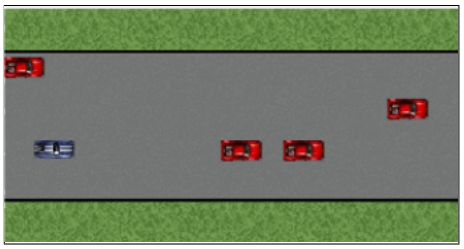

Fig. 5. The car driving domain. The blue car is the learning agent which could be tasked with avoiding the red cars.

For this domain, we generated two sets of trajectories, of length 128 and 500, with a 2% chance of switching reward functions at each timestep.

These generated trajectories were segmented using the NPBRS, subgoal state and linear dynamical systems (Labelled LDS) samplers.

Figure 4 presents a typical result from this process. The skills produced are labelled based on their correspondence

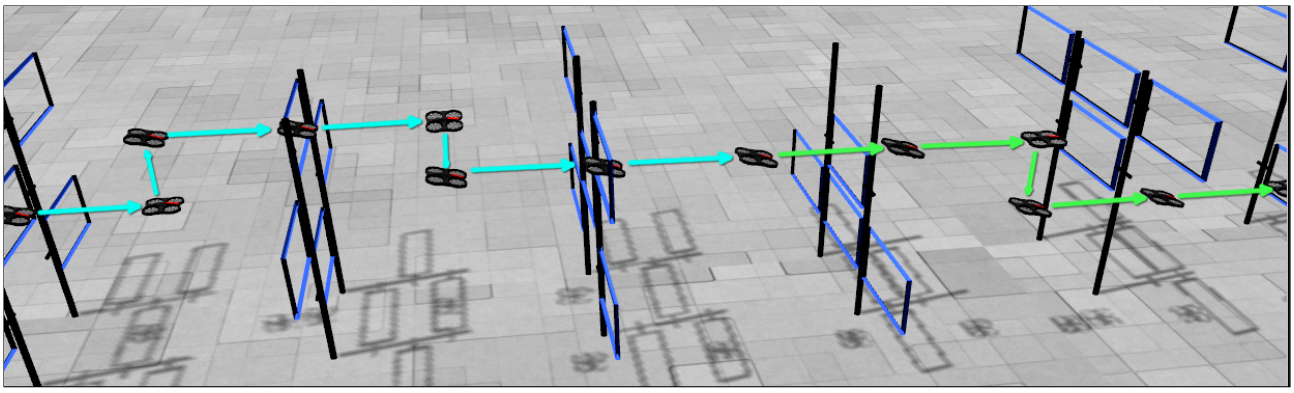

Fig. 6. The quadcopter gate domain. The agent is initially flying through as many hoops as possible (cyan) but switches to avoiding the hoops (green).

to skills in the true sequence. If a skill overlaps with a true skill for more than 50% of its occurrences, it is given the same letter. Thus, inferred skills B, B_1 and B_2 all received the label B because they mostly align with skill B in the true skill sequence across all trajectories. Skills which did not share more than 50% of their occurrences with any true skill are shown in grey.

NPBRS produces segmentations very similar to the true mode sequence. It makes small errors at the boundaries between skills because the skills agree on their action selections in these small windows, making the skill boundary indistinguishable.

The poor performance of the LDS sampler is expected as the method is not meant to model a system whose mode specific emissions are not a linear dynamical system. It is included here only to verify that different models are necessary in some domains, but does not indicate poor performance in more appropriate domains.

	Expert	NPBRS	Subgoal	IRL
Reward	39093	37832	-5660	-14453
Skill Switches	7.13	9.2	45.11	0
Skills	3	7.8	3.4	1

Fig. 7. Results from NPBRS, Subgoal state and unsegmented IRL in the 500 length highway domain averaged over 10 runs. Rewards and skill switches are presented as an average per trajectory

In Figure 7 we present aggregate results of using the inferred skills in the original trajectories according to the corresponding inferred segmentation. Since we are using these skills for control, we exclude LDS from the comparison, as the model is uncontrolled. We instead compare against unsegmented IRL. NPBRS achieves a much higher average reward than subgoal state segmentation or IRL, and is very close to that achieved by the original expert. It also switches skills at a similar frequency to the expert. It does, however, find 8 skills where the expert displayed 3, finding multiple skills corresponding to each expert skill. The subgoal state method has a negative average reward and switches skills frequently, indicating that it has not managed to find appropriate skills. Unsegmented IRL performs poorly in terms of reward received, meaning that it could not find a single reward function that explains the expert behaviour.

C. Quadcopter Gate Domain

In the quadcopter gate domain, a simulated quadcopter must make its way through an obstacle course. This was implemented using the TUM simulator [24], a 3D simulation of the AR.Drone quadcopter built in Gazebo [25] and integrated with ROS [26].

The obstacles involved are square hoops placed in a corridor, as shown in Figure 6. The quadcopter is able to control its motion in the height and width dimensions using the actions up, down, left and right, and moves forward by a fixed amount at every timestep. The corridor is discretized into a $3 \times 3 \times 500$ grid, with each 3×3 slice having one of 6 hoop configurations placed in it. The available hoop configurations are shown in Figure 8.

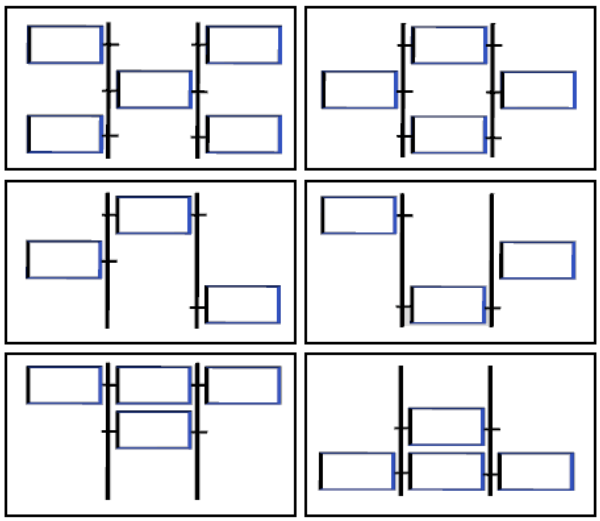

Fig. 8. The hoop configurations available in the quadcopter gate domain.

The state is described by the coordinates of the quadcopter

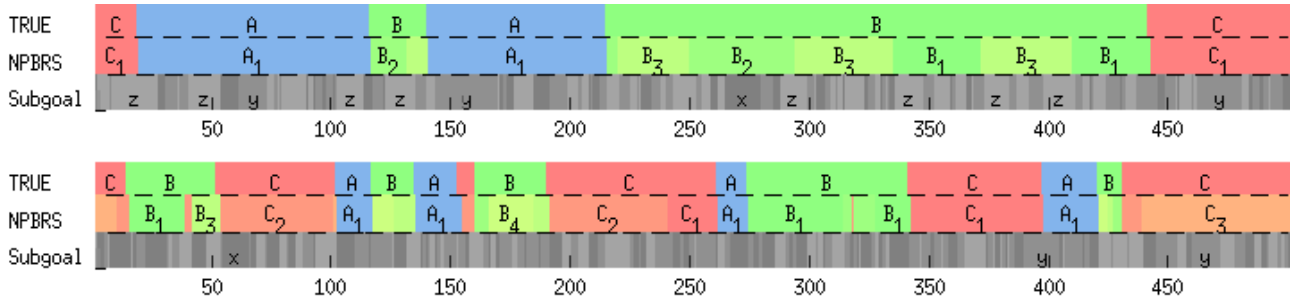

Fig. 9. Segmentations for the quadcopter gate domain for trajectory length 500 showing using segmentation based on NPBRS and subgoal states.

within the slice and the hoop configurations of the next four slices. We have designed three reward functions in this domain:

- A: Fly through as many hoops as possible. The agent receives a reward of 500 for flying through a hoop and zero otherwise.
- B: Fly as close to the ceiling as possible, avoiding flying through hoops. The agent receives a reward of 10 for being in the top row, a reward of -500 for flying through a hoop, and zero otherwise.
- C: Fly as close to the ground as possible, avoiding flying through hoops. The agent receives a reward of 10 for being in the bottom row, a reward of -500 for flying through a hoop, and zero otherwise.

A typical segmentation for the quadcopter domain is shown in Figure 9. NPBRS again performs very well, with the segmentation approximating the true mode sequence. It does however sometimes find multiple skills where a single true skill appears, as in the case of B in Figure 9 which is split into B_1, B_2, and B_3. This happens because IRL is an ill-posed problem, with multiple reward functions able to explain a trajectory [11].

Subgoal states perform very poorly, which is possibly a result of the stochasticity of the domain. The agent has no control over the hoop configurations which will be observed as these are randomly generated. It is thus impossible for an agent to formulate an effective policy to reach a particular subgoal state.

	Expert	NPBRS	Subgoal	IRL
Reward	82616	79501	-30045	139
Skill Switches	6.81	9.47	88.95	0
Skills	3	8.2	2.6	1

Fig. 10. Results from NPBRS, Subgoal state and unsegmented IRL in the quadcopter gate domain averaged over 10 runs. Rewards and skill switches are presented as an average per trajectory

Figure 10 presents aggregate results. Again, NPBRS achieves an average reward which is similar to the expert. Unsegmented IRL performed reasonably well in this domain, achieving a positive reward, and outperformed the subgoal state method by a large margin, indicating that the expert's behaviour in this domain is better modelled by a complex reward function rather than a combination of simple ones. NPBRS greatly outperforms both unsegmented IRL and the subgoal state method. It also indicates that the method is not data intensive, as it was able to model expert behaviour using only 16 input trajectories.

V. RELATED WORK

Surana and Srivastava [27] provide a nonparametric Bayesian method for recovering multiple reward functions from a single set of trajectories. They use the HDP-HMM formulation. This requires the skill transition dynamics to be consistent across all of the trajectories, which fails in the case where a library of skills is used in trajectories with different goals [18]. The formulation thus discourages shared skills across trajectories.

Niekum et al. [16] extend the BP-AR-HMM by post-processing the discovered segments, learning Dynamic Movement Primitives which provide better generalization and the ability to use RL for policy improvement. This is performed after segmentation on the basis of linear dynamics and is hence limited by the quality of the initial segmentation.

Clustering methods exist to discover multiple reward functions from unlabelled demonstration trajectories where each trajectory is assumed to be generated from a single reward function [28], [29]. The methods cluster trajectories based on their reward functions, defining IRL methods to do so. They are unsuitable for situations where each trajectory is made up of multiple skills as they do not attempt to perform trajectory segmentation.

VI. SUMMARY

We have presented a method that segments unstructured demonstrations based on the skill being employed at each timestep, with skills represented by reward functions in an MDP. We have demonstrated that this method is better suited to some domains than representing skills as linear dynamical systems or subgoal states.

ACKNOWLEDGMENT

The authors would like to thank Emily Fox, Michael C. Hughes and Sergey Levine for making their well structured code freely available.

REFERENCES

[1] J. Nakanishi, J. Morimoto, G. Endo, G. Cheng, S. Schaal, and M. Kawato, "Learning from demonstration and adaptation of biped locomotion," *Robotics and Autonomous Systems*, vol. 47, no. 2, pp. 79–91, 2004.

[2] P. Abbeel, A. Coates, and A. Ng, "Autonomous helicopter aerobatics through apprenticeship learning," *The International Journal of Robotics Research*, vol. 29, no. 13, pp. 1608–1639.

[3] B. G. Weber, M. Mateas, and A. Jhala, "Learning from demonstration for goal-driven autonomy." in *Proceedings of the 26th AAAI Conference on Artificial Intelligence*, 2012.

[4] B. J. Michini, "Bayesian nonparametric reward learning from demonstration," Ph.D. dissertation, Massachusetts Institute of Technology, 2013.

[5] R. Sutton and A. Barto, *Introduction to reinforcement learning*. MIT Press, 1998.

[6] A. Ng and S. Russell, "Algorithms for inverse reinforcement learning." in *Proceedings of the 17th International Conference on Machine Learning,*, 2000, pp. 663–670.

[7] G. Konidaris, S. Kuindersma, R. Grupen, and A. Barto, "Robot learning from demonstration by constructing skill trees," *International Journal of Robotics Research*, vol. 31, no. 3, pp. 360–375, March 2012.

[8] S. Niekum and A. G. Barto, "Clustering via Dirichlet process mixture models for portable skill discovery," in *Advances in neural information processing systems*, 2011, pp. 1818–1826.

[9] E. Fox, M. Hughes, E. Sudderth, and M. Jordan, "Joint modeling of multiple time series via the beta process with application to motion capture segmentation," *The Annals of Applied Statistics*, 2014.

[10] R. Sutton, D. Precup, and S. Singh, "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning," *Artificial Intelligence*, vol. 112, pp. 181–211, 1999.

[11] P. Abbeel and A. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the 21st International Conference on Machine Learning*, 2004.

[12] S. Levine, Z. Popovic, and V. Koltun, "Nonlinear inverse reinforcement learning with gaussian processes," in *Advances in Neural Information Processing Systems*, 2011, pp. 19–27.

[13] A. Ng and S. Russell, "Algorithms for inverse reinforcement learning," in *Proceedings of the 17th International Conference on Machine Learning*, 2000, pp. 663–670.

[14] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning." in *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, 2008, pp. 1433–1438.

[15] S. Chiappa and J. R. Peters, "Movement extraction by detecting dynamics switches and repetitions," in *Advances in Neural Information Processing Systems*, 2010, pp. 388–396.

[16] S. Niekum, S. Osentoski, G. Konidaris, and A. G. Barto, "Learning and generalization of complex tasks from unstructured demonstrations," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 5239–5246.

[17] M. Alvarez, J. R. Peters, N. D. Lawrence, and B. Schölkopf, "Switched latent force models for movement segmentation," in *Advances in Neural Information Processing Systems*, 2010, pp. 55–63.

[18] E. Fox, "Bayesian nonparametric learning of complex dynamical phenomena," Ph.D. Thesis, MIT, Cambridge, MA, 2009.

[19] M. C. Hughes, E. B. Sudderth, and E. B. Fox, "Effective split-merge Monte Carlo methods for nonparametric models of sequential data," in *Advances in Neural Information Processing Systems*, 2012, pp. 1295–1303.

[20] B. Michini and J. P. How, "Bayesian nonparametric inverse reinforcement learning," in *Machine Learning and Knowledge Discovery in Databases*, 2012, pp. 148–163.

[21] B. Michini, M. Cutler, and J. P. How, "Scalable reward learning from demonstration," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 303–308.

[22] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[23] C. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[24] J. Engel, J. Sturm, and D. Cremers, "Camera-based navigation of a low-cost quadrocopter," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 2815–2821.

[25] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 3. IEEE, pp. 2149–2154.

[26] Robot operating system. [Online]. Available: http://www.ros.org

[27] A. Surana and K. Srivastava, "Bayesian nonparametric inverse reinforcement learning for switched Markov decision processes," in *Proceedings of the 13th International Conference on Machine Learning and Applications*, 2014, pp. 47–54.

[28] J. Choi and K.-E. Kim, "Nonparametric Bayesian inverse reinforcement learning for multiple reward functions," in *Advances in Neural Information Processing Systems*, 2012, pp. 305–313.

[29] M. Babes, V. Marivate, K. Subramanian, and M. L. Littman, "Apprenticeship learning about multiple intentions," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 2011, pp. 897–904.