Path planning with user route preference - A reward surface approximation approach using orthogonal Legendre polynomials

Aravinda Ramakrishnan Srinivasan, and Subhadeep Chakraborty *Member, IEEE*

Abstract— **As self driving cars become more ubiquitous, users would look for natural ways of informing the car AI about their personal choice of routes. This choice is not always dictated by straightforward logic such as shortest distance or shortest time, and can be influenced by hidden factors, such as comfort and familiarity. This paper presents a path learning algorithm for such applications, where from limited positive demonstrations, an autonomous agent learns the user's path preference and honors that choice in its route planning, but has the capability to adopt alternate routes, if the original choice(s) become impractical. The learning problem is modeled as a Markov decision process. The states (way-points) and actions (to move from one way-point to another) are pre-defined according to the existing network of paths between the origin and destination and the user's demonstration is assumed to be a sample of the preferred path. The underlying reward function which captures the essence of the demonstration is computed using an inverse reinforcement learning algorithm and from that the entire path mirroring the expert's demonstration is extracted. To alleviate the problem of state space explosion when dealing with a large state space, the reward function is approximated using a set of orthogonal polynomial basis functions with a fixed number of coefficients regardless of the size of the state space. A six fold reduction in total learning time is achieved compared to using simple basis functions, that has dimensionality equal to the number of distinct states.**

I. INTRODUCTION

With the advent of autonomous cars, there is need for a more user-centric path planning paradigm, one that is capable of incorporating user route preferences into the planning algorithm in a natural way, while still accommodating for external factors which might necessitate a deviation from the preferred route. A human driver takes into account several criteria subconsciously and it is often impossible and probably cumbersome for the individual to account for, list and quantify all the factors which motivate a particular decision. It is simpler and more natural to provide a positive demonstration of the preferred routes. The agent learns the implicit policy for which the demonstrated route is optimal and then implements the same when it is tasked to plan and execute the path the next time. In a sense, our vision is of an autonomous car/agent functioning as an apprentice to the human driver. Just like an apprentice, the agent first learns the user preferred path from a demonstration. It can then extrapolate the known solution space to find alternate path, if the original solution is rendered unusable due to changed circumstances. Essentially, the agent has to be capable of automated path planning/executing while taking into account user preferences.

A.R.K. Srinivasan and S. Chakraborty are with the Department of Mechanical Aerospace and Biomedical Engineering, University of Tennessee, Knoxville, TN 37996 USA
email: asriniv2@vols.utk.edu
email: schakrab@utk.edu

Autonomous path planning has a long and illustrious history - algorithms can be traced back to Dijkstra algorithm [1] [2], used to find the shortest path from an origin node to destination node in a connected graph. This was followed by the goal directed search algorithm (A*) [3]. This incorporated formal methods to modify the weight of the edges in the connected graph to achieve quicker path planning. More recent development tries to incorporate the time dependency into route planning agent [4]. The goal is to take into account the departure time to find the minimum travel time to destination. Kriegel et.al. [5] tried to take user's preference into route planning. This is based on the skyline operator for searching a database to rank various results according to user preferences. Here the user preferences has to be explicitly specified in order to find the desired route. The work presented in this paper takes into account the user preferences implicitly from the demonstration provided by the user.

Learning from demonstration is an interesting paradigm usually studied in the robotics context and has been tackled by many researchers. Most of the prior work have tried to address it from the viewpoint of database building and searching in the database for the current situation and executing the script from the database [6]. Initiated in late 1980s as imitation learning, the target of early research in reinforcement learning was to make manipulators follow similar path from start to goal as previously demonstrated by an expert. Segre and Dejong [7] extracted a set of 'if-then' sequences to achieve the path imitation. Given the limitations of available computing resources in the late $80's$, this itself was a compelling feat. As the computing power and sensor technology continued to improve, researcher began to develop systems that are more intelligent. Latest imitation learning technique as reported in [8] tries to incorporate both position and force profile into the learning domain. Another work [9] tries to use Gaussian Mixture Model and Gaussian Mixture Regression to learn the way-points to either lead/follow in the task of picking up an object alongside a human. Also a recent work by Billard's group trained a manipulator both in simulation and in real-time to catch a flying object [10], [11].

Another body of work by Veloso's group introduced a new method called confidence based autonomy [12], [13]. The basic building block of their algorithm was a robust database where each distinct state action pair is stored. Thus, whenever in real time execution, a state is encountered, the agent queries the database for a suitable action which then returns a suitable action along with a confidence parameter.

Each of the techniques for learning from demonstration described above has its own unique advantages and disadvantages. The problem with database-oriented technique is the storage of all the relevant information from training in an intelligent manner for it to be quickly accessible. If the information becomes too large then real time fetching will become time consuming. There is a similar state space explosion problem associated with Markov decision processes. The time to find the optimal solution scales exponentially with the number of distinct states.

There has been a body of work by Ng's group [14]–[16] on modeling the learning problem as a Markovian process. The demonstrations are assumed to be executed according to an expert's policy, which is considered as the optimal solution to the implicit Markov Decision Process (MDP) with unknown reward functions. The inverse reinforcement learning algorithm is used to compute the unknown reward function from the expert's demonstration(s). In the work by Kim et.al. [17], [18], the path planning with human input is accomplished by hand-picking a set of features and learning the weights for each feature by using inverse reinforcement learning. Similarly [19] attempts to incorporate human factor into autonomous path planning by selecting specific features from the sensor input. The pros and cons of different feature sets are dealt with in [20]. The failed set of demonstration were used in [21]. Nguyen et. al. [22] splits the state space into different region and computed the augmented reward function by utilizing expectation maximization technique. Ziebart et.al. [23] utilized maximum entropy method to learn and predict user's route preference and destination. There is also a work by Deisenroth and et.al. [24] wherein they try to account for incomplete models. In all of these works, domain expertise is required in order to hand pick the feature set.

In this work, we are also trying to model the agent as an MDP with unknown reward functions to be learned from demonstration(s). The difference from the previous work is that we are trying to circumvent the need for domain knowledge and hand picking the feature set by utilizing the orthogonal polynomial functions as basis functions for representing the reward structure(the feature set). Additionally, we can circumvent the problem of state space explosion by utilizing polynomial function of order lower than that of the state space. This is largely inspired by image reconstruction techniques employed in image processing community [25].

Considering the difficulties in implementing a real autonomous vehicle, for proof of concept, the AI agent utilized in this paper is an autonomous mobile robot learning the operator's preference for a particular route through reward/inverse reinforcement learning. Section II discusses in detail about the theoretical principle underlying the inverse reinforcement learning algorithm. Section III explains the experimental setup and Section IV elucidates the results obtained from real-time experiments.

II. INVERSE REINFORCEMENT LEARNING

This paper employs two underlying principles, namely Markov decision process [26], [27] and inverse reinforcement learning [14]–[16] developed by Ng's group. A very succinct description of the algorithm is provided here for clarity and completeness.

A Markov decision process $M = \{S, A, P_{(sa)}, \gamma, R\}$ consists of the following

- S Set of all possible states of the system.
- A Set of actions available to the system.
- P Transition probability $\mathrm{P}(s, a, s')$ which gives the probability of transition to state s' from state s by taking action a.
- R Set of rewards - This indicates the payoff from the various states of the system. The system's overall behavior depends on the rewards.
- γ Discount factor $\in [0, 1)$ - This parameter controls the relative weights of rewards acquired in near vs. distant future.

The basic underlying assumption of Markov decision process is that the current state and the action taken alone determine the next state, independent of past states or actions. For an MDP, the policy denoted as π is a prescription of actions to be taken in given states. A policy is optimal with respect to maximizing the cumulative discounted future rewards, if it satisfies the Bellman optimality equation. To describe the optimality equation, $\forall s \in S$ and $a \in A$, the value function V^π and Q function Q^π have to satisfy

$$V^\pi(s) = R(s) + \gamma \Sigma_{s'} \left(P_{s\pi(s)}(s') \right) V^\pi(s') \tag{1}$$

$$Q^\pi(s, a) = R(s) + \gamma \Sigma_{s'} \left(P_{sa}(s') \right) V^\pi(s') \tag{2}$$

The value function and Q function represent the expected cumulative reward for following the given policy π and a policy π is an optimal policy π^* for M if and only if $\forall s \in S$,

$$\pi(s) \in arg\ max_{a \in A} Q^\pi(s, a) \tag{3}$$

This simply states that at any given state, the action chosen must result in the system being in the best possible next state with respect to their calculated value.

In the inverse problem, the agent does not have direct access to the underlying reward function, but is only shown positive examples of how a task might be performed. The assumption is that the demonstrator has an implicit reward function and the demonstration is a manifestation of the optimal policy with respect to that reward function. The inverse reinforcement learning problem deals with extracting the reward function that best explains the policy demonstrated by the expert.

We restrict ourselves to the case of $S = \mathbb{R}^2$, for example, longitude and latitude can completely specify intersections. If we consider the state space to be 2-dimensional then the reward function computed by the inverse reinforcement learning algorithm has to map from $\mathbb{R}^2 \longrightarrow \mathbb{R}$. Considering

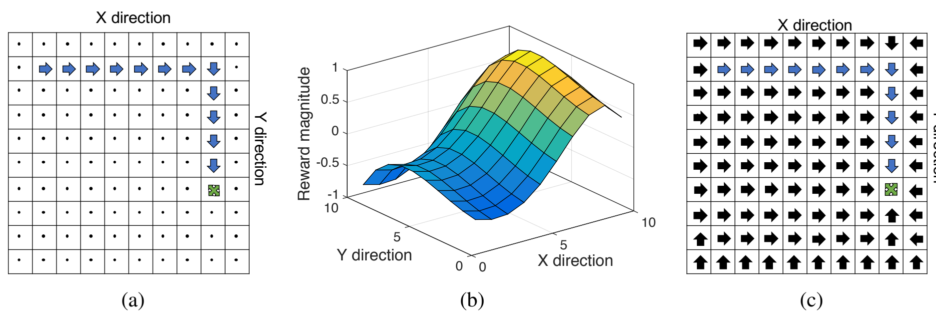


Fig. 1. (a) The path demonstrated to the Turtlebot, (b) The extracted reward for the path and, (c) The optimal policy extracted from the reward function

the difficulty of optimizing over this space, a linear approximation for the reward function can be used, where

$$R(s) = \alpha_1\phi_1(s) + \alpha_2\phi_2(s) + \alpha_3\phi_3(s) + \ldots \alpha_n\phi_n(s) \tag{4}$$

In [14] [16], for the linear approximation of the reward function, R the expert's had hand picked the feature set. This will make it similar to existing techniques for user to input their route preference [17]–[19], [22]. However, if no such insight is available, a simple but impractical set of basis functions with the same dimensionality as the number of states can be constructed as follows. For instance, an example basis function array for a space discretized into $2 \times 2 = 4$ distinct states can be

$$\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \text{and} \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$$

where each matrix represent one of the basis function. This is the simplest of basis function array which can represent any reward function shape in 2D for the 2×2 state space. But it is evident that with increasing number of states this will lead to exponential increase in computation time for the inverse algorithm. To alleviate the problem, we take inspiration from the image processing community [25], where multivariate orthogonal polynomials are used as basis functions to find the image moments. One discrete orthogonal polynomial function that has been tested with success is Legendre polynomial of different orders. A Legendre polynomial is given by

$$P_n(x) = \frac{1}{2^n n!}\frac{d^n}{dx^n}\left[(x^2-1)^n\right] \tag{5}$$

where n denotes the order of the polynomial.

The reward function is actually a complex envelope encompassing the entire state space and to find the equations governing that envelope, utilizing a set of orthogonal polynomials reduces the number of variables to be optimized. The only variables that need to be optimized are a fixed number of coefficients of the orthogonal polynomials, regardless of the size of the state space. The orthogonality of the polynomial function allows to compute the coefficients for each dimensions separately and then use tensor product to find the value for a given (x,y) coordinate. This is evident from the reward envelope (shown in Fig. 1(b)) found by the modified algorithm for autonomous robot navigation. The smooth surface of the reward function is the result of using the weighted sum of orthogonal polynomial basis function to approximate the original implicit reward function. If we approximate the reward function, R with Legendre polynomials, then R is given by

$$R(s) = \alpha_1\phi_1\theta_1 + \alpha_2\phi_1\theta_2 + \alpha_3\phi_1\theta_3 + \ldots \alpha_{n\times n}\phi_n\theta_n \tag{6}$$

where n is order of Legendre polynomial and θ and ϕ are the Legendre polynomials of various orders, one for each dimension. The α_i are the parameter our inverse reinforcement learning algorithm is trying to compute. Since expectation is a linear function, the value function, V for the reward function, R given by equation (6) is

$$V^\pi = \alpha_1 V_1^\pi + \alpha_2 V_2^\pi + \cdots + \alpha_{n\times n} V_{n\times n}^\pi \tag{7}$$

Thus Bellman's optimality equation (3) can be written as

$$E_{s'\sim P_{sa_1}}[V^\pi(s')] \geq E_{s'\sim P_{sa}}[V^\pi(s')] \tag{8}$$

for all states s and all actions $a \in A \setminus a_1$. This merely states the Bellman equation (3) in another form. From equation (7), we know that $V^\pi(s)$ is a linear combination of basis function weighted by α_i. Hence we can formulate the problem as linear programming (LP) to find the constraints (α_i).

We utilize the linear programming formulation from Ng and et.al. work [16]

$$\begin{aligned} &\text{maximize} \sum_{s\in S_0} \min_{a\in\{a_2,\ldots,a_{n\times n}\}}\{ \\ &p(E_{s'\sim P_{sa_1}}[V^\pi(s')] - E_{s'\sim P_{sa}}[V^\pi(s')])\} \\ &s.t. |\alpha_i| \leq 1, i = 1,\ldots,n\times n \end{aligned} \tag{9}$$

The α_i comes into play through (7) and the penalty function used here is given by $p(x) = x$ if $x \geq 0$, $p(x) = 2x$ otherwise.

Thus, the current algorithm developed in [14] [16] and modified to suit the tele-operated system(s) is as follows

- Step 1: Initialize with a set of basis functions. *A set of Legendre polynomial with fixed order is chosen in this paper.*
- Step 2: Calculate the value of the states using value iteration algorithm for the expert's policy.
- Step 3: Randomly pick a policy and add it to set of policies. (A random policy is used to seed the algorithm)

- Step 4: Calculate the value of the states using value iteration algorithm with each of the basis function for all the policies in the set.
- Step 5: Maximize the weighted difference between the expert's policy value and the average value from the set of policies.
- Step 6: Use the maximized weight to find a reward function.
- Step 7: Utilize the Q-Value, find the respective policy for the reward function, and add it to the set containing the random initial policy.
- Step 8: Run step 4 through 7 until a reward function satisfying the expert's policy is obtained.

The weighted difference between expert's policy and average value from the all other policies in the set is maximized, in a sense we are trying to find a reward function that maximally differentiates between expert's policy and all other possible policies. The extracted weight/reward function can be utilized to find the complete policy of the expert. The order of the polynomial is found by starting with order 2 and increasing in steps of 1 till a sufficient representation of reward function is achieved. In our test case with 100 distinct states in 2D space, a pair of Legendre polynomial with order 6 was sufficient to find reward function for all of the test paths. Thus instead of a maximization problem posed over 100 coefficients, it is reduced to only 36 (6×6) coefficients. Thus we circumvent the state space explosion problem by utilizing orthogonal polynomials of an order much lower in comparison to the number of distinct states in the system.

III. EXPERIMENTAL SETUP

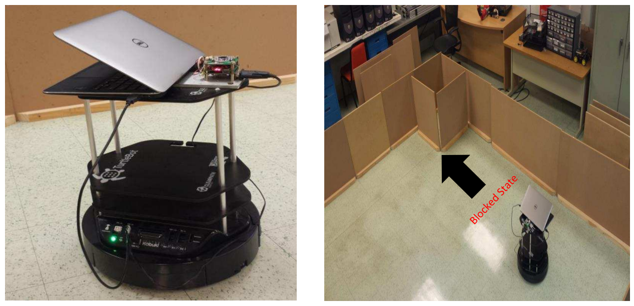
The experimental setup for the path-planning robot consist of a Turtlebot and a stargazer indoor GPS system. The Turtlebot is a low cost robot kit which runs on open source software ROS (Robot Operating System). The stargazer is a low cost indoor GPS which works on the principle of infrared image processing. Markers on the ceiling are read by an infrared camera on the stargazer and analyzed on board to provide the estimates of current position and orientation for the Turtlebot.

A point to be noted is the data from the stargazer is prone to noise. The same has manifested itself as random points in the reconstructed path. Also the stargazer sensor has been mounted off-center on the Turtlebot (figure 2(a)) which has lead to small loops in the reconstructed trajectories whenever the Turtlebot was making turns. The work flow can be simply stated as follows. First, a demonstration from an expert is recorded. The state space is divided into rectangular grids and from the recorded demonstration the state-actions pair are interpreted. Then the modified inverse reinforcement learning algorithm is run on the available data and once a suitable expert policy is extracted, the algorithm is stopped and the policy is fed back to the autonomous agent.

- The first experiment was designed to show that the Turtlebot can acquire the human demonstrated path and follow the same in the autonomous mode. The state space has been defined as equal sized square on the arena floor and for the current experiment a total of 25 states were utilized. The action for the Turtlebot are restricted to rotate left, rotate right, move forward, move backward and halt. Once a demonstration is recorded the GPS data are utilized to extract the states and the state transition in the demonstrated path. Then the modified inverse reinforcement learning algorithm is run and the expert's unknown reward function and the complete policy is calculated.
  - As a next step, a corner that comes in the path is cordoned off and the ability of the algorithm to come up with an alternate policy which matches the expert's path as much as physically possible is tested. For this step, the state transition into the blocked corner is voided.
- The next experiment is to demonstrate a complex path to the Turtlebot and then once a policy is extracted by the algorithm, the Turtlebot is started from a different start point to test the ability of the robot to still follow the expert's demonstrated path. This experiment was to show the ability of the algorithm to extract a reward function for a complex policy and also reach the destination from a different start point and match the expert's policy in an intelligent way.
- The last set of experiments is done to show the advantage of utilizing the polynomial basis function. For this, the complex path (path with maximum number of permissible turns) is taken. The learning algorithm is run for different number of distinct states with both the simple basis function set (has dimensionality equal to the number of states) and polynomial basis function (fixed number of coefficients regardless of the number of states). The time complexity graphs showing the results are generated.

IV. RESULTS AND DISCUSSION

Figure 3 shows the path demonstrated by an expert to the Turtlebot (in blue). The path followed in autonomous mode after the policy is extracted using the inverse reinforcement learning algorithm is similar to the demonstrated path, thus validating that the extracted policy tries to mirror expert's



(a) (b)

Fig. 2. (a) The Turtlebot platform equipped with a Stargazer indoor GPS. (b) Turtlebot in the arena. A corner in the arena is blocked to test the ability to adaptively re-plan.

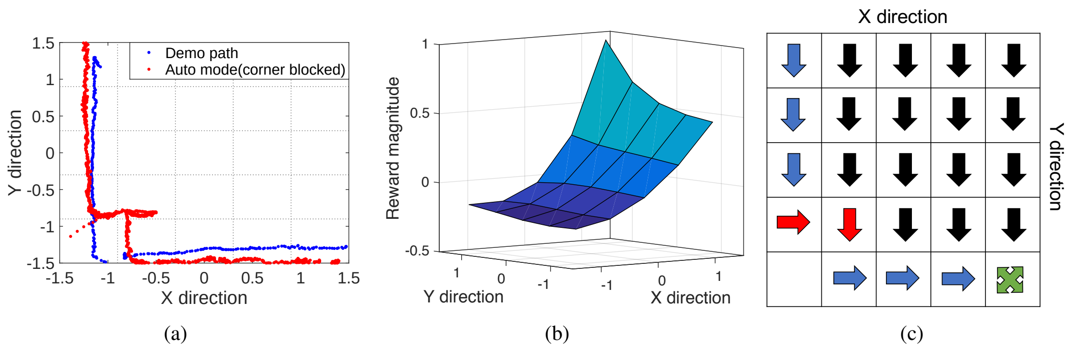


Fig. 3. (a) The demonstrated path (blue) and the path followed by the Turtlebot in autonomous mode (red) when a corner in the demonstrated path is made inaccessible, (b) The extracted reward for the demonstrated path and, (c) The optimal policy extracted from the reward function

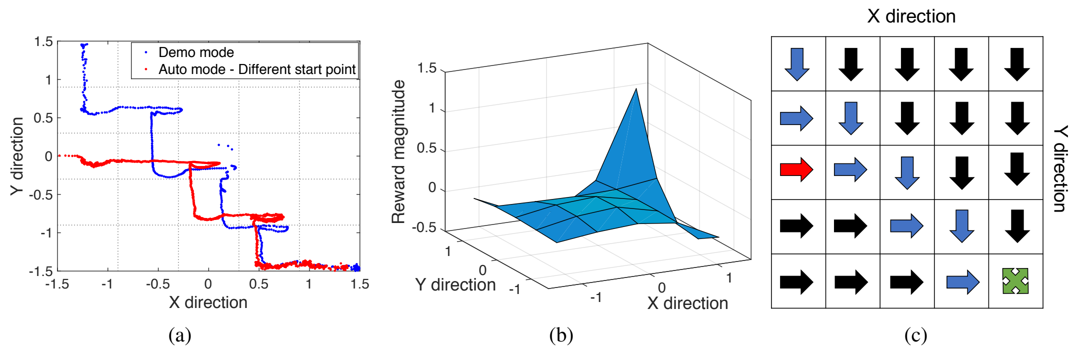


Fig. 4. (a) The path demonstrated and followed from a different starting point by the Turtlebot, (b) The extracted reward for the path and, (c) The optimal policy extracted from the reward function

path. The arrow in the policy graph corresponds to the desired direction of movement as extracted by the algorithm. Figure 3 shows the ability of the robot to maneuver the cordoned off corner and follow the expert's path as much as physically possible (shown in red).

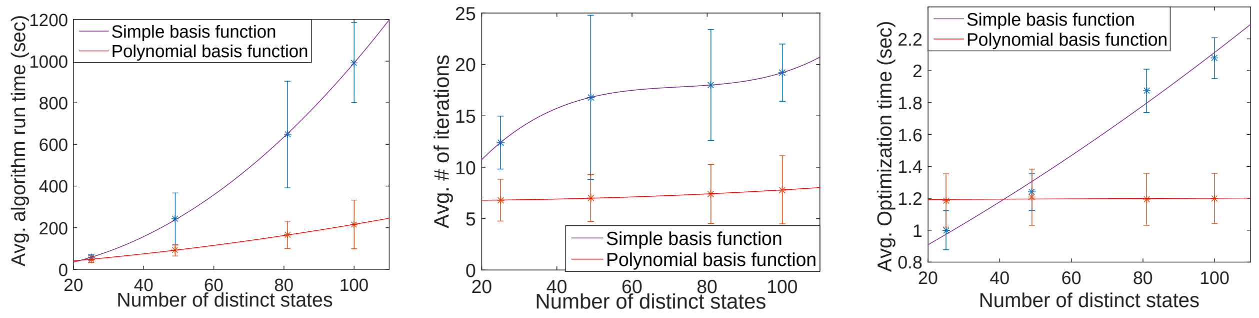
Figure 4 shows the ability of the robot to follow even a complex path from a different starting point. It may be noted that the learned policy tries to keep to as much of the demonstrated path as possible. In other words, even from a different starting point, the robot joins the demonstrated path as quickly as it can without violating any physical constraints.

Figure 5 shows required computation times for different number of distinct states. Figure 5(a) shows that the average time to run the complete learning algorithm with the simple basis function increases exponential with the number of distinct state. Whereas the average run time with the polynomial basis function is almost linear with the number of distinct state. This is result of constant number of variables to be optimized in case of the polynomial basis function compared to increasing number of optimization variables in case of the simple basis function. The linear increase in polynomial basis function case is the result of running value iteration for increased number of states. Figure 5(b) shows the average number of iterations required for the algorithm to find the expert's implicit reward function. Figure 5(c) depicts the time taken by just the optimization routine to find the solution for given set. Since the number of optimization variables is constant in the polynomial basis function case, the optimization routine time does not change with increasing number of distinct states. But, in the case of simple basis function, the optimization routine time increases exponentially with number of distinct states and thus results in more run time for the entire learning algorithm. Time is a crucial factor when running real time systems and the graphs prove that it is advantageous to approximate the reward function using polynomial basis functions.

V. CONCLUSIONS

Thus the expert/user can provide a demonstration to the agent, which is more natural than specifying the user preferences. From that demonstration the underlying implicit reward function for the user preferences can be extracted in a timely manner and utilized to autonomously run the agent. The mental load on the user to explicitly specify their preferences over the entire state space is removed and the user can interact with the AI and provide the necessary input in an intuitive and easy manner. The path planning with the Turtlebot is a proof of concept experiment to show the viability of the algorithm to work with larger state space.

From a broader perspective, the results prove that this research is an important step towards better human-robot collaboration. If the robot has the ability to learn from a human demonstration, then the human can start to feel that the robot is like a coworker. A robot that has learning from demonstration capability can help in forgoing the need to



(a) (b) (c)

Fig. 5. (a) Number of distinct states vs. average time taken for the learning algorithm to find the expert's reward function, (b) Average number of iterations taken by the algorithm to find the expert's reward function, (c) The average time taken for the optimizer to find a solution

learn a special language to code a new task in manufacturing industry. An intelligent rescue robot will be able to plan according to the situation from previously learned plans. This will reduce the time to complete the mission, as there will be no need to plan from scratch. A robot that is capable of reporting the relevant data from a failure can reduce the time to debug. This is essentially a flexible communication framework. Additionally, if the natural language processing engine is included into the framework, it can bring the human-robot collaboration experience to the next level.

ACKNOWLEDGMENT

Subhadeep Chakraborty thanks the Startup funds provided by the Mechanical Aerospace and Biomedical Engineering Department at the University of Tennessee.

REFERENCES

[1] S. Skiena, "Dijkstra's algorithm," *Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica, Reading, MA: Addison-Wesley*, pp. 225–227, 1990.

[2] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959. [Online]. Available: http://dx.doi.org/10.1007/BF01386390

[3] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *Systems Science and Cybernetics, IEEE Transactions on*, vol. 4, no. 2, pp. 100–107, July 1968.

[4] D. Delling, P. Sanders, D. Schultes, and D. Wagner, "Engineering route planning algorithms," in *Algorithmics of large and complex networks*. Springer, 2009, pp. 117–139.

[5] H.-P. Kriegel, M. Renz, and M. Schubert, "Route skyline queries: A multi-preference path planning approach," in *Data Engineering (ICDE), 2010 IEEE 26th International Conference on*. IEEE, 2010, pp. 261–272.

[6] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[7] A. M. Segre and G. DeJong, "Explanation-based manipulator learning: Acquisition of planning ability through observation," in *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, vol. 2. IEEE, 1985, pp. 555–560.

[8] P. Kormushev, S. Calinon, and D. G. Caldwell, "Imitation learning of positional and force skills demonstrated via kinesthetic teaching and haptic input," *Advanced Robotics*, vol. 25, no. 5, pp. 581–603, 2011.

[9] P. Evrard, E. Gribovskaya, S. Calinon, A. Billard, and A. Kheddar, "Teaching physical collaborative tasks: Object-lifting case study with a humanoid," in *Humanoid Robots, 2009. Humanoids 2009. 9th IEEE-RAS International Conference on*. IEEE, 2009, pp. 399–404.

[10] S. Kim, A. Shukla, and A. Billard, "Catching objects in flight," *Robotics, IEEE Transactions on*, vol. 30, no. 5, pp. 1049–1065, 2014.

[11] S. Kim and A. Billard, "Estimating the non-linear dynamics of free-flying objects," *Robotics and Autonomous Systems*, vol. 60, no. 9, pp. 1108–1122, 2012.

[12] S. Chernova and M. Veloso, "Interactive policy learning through confidence-based autonomy," *Journal of Artificial Intelligence Research*, vol. 34, no. 1, p. 1, 2009.

[13] ——, "Multi-thresholded approach to demonstration selection for interactive robot learning," in *Human-Robot Interaction (HRI), 2008 3rd ACM/IEEE International Conference on*. IEEE, 2008, pp. 225–232.

[14] 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.

[15] ——, "Exploration and apprenticeship learning in reinforcement learning," in *Proceedings of the 22nd international conference on Machine learning*. ACM, 2005, pp. 1–8.

[16] A. Y. Ng, S. J. Russell *et al.*, "Algorithms for inverse reinforcement learning." in *Icml*, 2000, pp. 663–670.

[17] B. Kim and J. Pineau, "Socially adaptive path planning in human environments using inverse reinforcement learning," *International Journal of Social Robotics*, pp. 1–16, 2015.

[18] ——, "Human-like navigation: Socially adaptive path planning in dynamic environments," in *RSS 2013 Workshop on Inverse Optimal Control and Robotic Learning from Demonstration*, 2013.

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

[20] D. Vasquez, B. Okal, and K. O. Arras, "Inverse reinforcement learning algorithms and features for robot navigation in crowds: an experimental comparison," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE, 2014, pp. 1341–1346.

[21] K. Shiarlis, J. Messias, M. van Someren, and S. Whiteson, "Inverse reinforcement learning from failure," in *RSS 2015: Proceedings of the 2015 Robotics: Science and Systems Conference, Workshop on Learning from Demonstration: Inverse Optimal Control, Reinforcement Learning, and Lifelong Learning*, July 2015.

[22] Q. P. Nguyen, B. K. H. Low, and P. Jaillet, "Inverse reinforcement learning with locally consistent reward functions," in *Advances in Neural Information Processing Systems*, 2015, pp. 1738–1746.

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

[24] M. Deisenroth and C. E. Rasmussen, "Pilco: A model-based and data-efficient approach to policy search," in *Proceedings of the 28th International Conference on machine learning (ICML-11)*, 2011, pp. 465–472.

[25] H. Zhu, "Image representation using separable two-dimensional continuous and discrete orthogonal moments," *Pattern Recognition*, vol. 45, no. 4, pp. 1540–1558, 2012.

[26] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[27] R. Bellman, "The theory of dynamic programming," DTIC Document, Tech. Rep., 1954.