Dynamic Obstacle Avoidance with PEARL: PrEference Appraisal Reinforcement Learning

Aleksandra Faust[1], Hao-Tien Chiang[1], Nathanael Rackley[1] and Lydia Tapia[1]

Abstract— **Manual derivation of optimal robot motions for task completion is difficult, especially when a robot is required to balance its actions between opposing preferences. One solution has been to automatically learn near optimal motions with Reinforcement Learning (RL). This has been successful for several tasks including swing-free UAV flight, table tennis, and autonomous driving. However, high-dimensional problems remain a challenge. We address this dimensionality constraint with PrEference Appraisal Reinforcement Learning (PEARL), which solves tasks with opposing preferences for acceleration controlled robots. PEARL projects the high dimensional continuous robot state space to a low dimensional preference feature space resulting in efficient and adaptable planning. We demonstrate that on a dynamic obstacle avoidance robotic task, a single learning on a much simpler problem performs real-time decision-making for significantly larger, high dimensional problems working in unbounded continuous states and actions. We trained the agent with 4 static obstacles, while the trained agent avoids up to 900 dynamic obstacles in a highly constrained space. We compare these tasks to traditional, often manually tuned solutions for these high-dimensional problems.**

I. Introduction

There are many high-dimensional, motion-based robotic tasks, including multi-robot coordination and control of complex kinematic linkages. These complex robotic problems often require planning high-dimensional motions that complete the task in a timely manner. Motion and trajectory planning identifies a sequence of actions that move the robot in accordance to its dynamics (physical constraints) and the task objectives. Since manually accounting for all possibilities is often infeasible, sampling-based, learning-based, and other intelligent methods are the norm [15]. Reinforcement learning (RL), in particular, has been successful for robotic task learning [12] in several problems such as table tennis [18], swing-free UAV delivery [5], and a self-driving car [9]. However, traditional RL methods do not handle continuous and high-dimensional state spaces well [8].

A primary challenge in these problems is task description. The task goals and constraints that the robot must obey are often unknown or difficult to calculate. For example, consider a simple manipulation task; a robot is required to set a glass on a table without breaking it. We do not know precisely the amount of force that causes the glass to shatter, yet we can describe our preferences: low force and fast task completion. *Preference reinforcement learning* learns and performs preference balancing tasks (PBTs) with desired qualities (preferences). These tasks cannot be easily demonstrated, but the set of preferences can be described [25].

We present PrEference Appraisal Reinforcement Learning (PEARL) for solving high-dimensional motion-based PBT problems (Figure 1). PEARL trains the planning agent on small problems, and transfers the learned policy to be used for planning on high-dimensional problems. The key to PEARL is the feature selection method that constructs task-preference features invariant to the robot's state space dimensionality. Because the method learns and performs the task in the feature space, such transfer is possible. Previously, we empirically showed that, using hand-crafted features, batch RL learns in small spaces and acts on larger problems [7], [6], but did not address when learning transfer is possible and how to do it for an arbitrary problem. This paper formalizes the feature selection and the conditions under which transfer is possible so that it can be applied for classes of PBTs. We include preferences that increase over time as well.

We demonstrate PEARL on the obstacle avoidance problem. The case study shows that the method is fast, easy to use, successful for high-dimensional problems, and is a useful tool in a RL-based motion planning toolbox. An agent is required to move to the goal without colliding with hundreds of stochastically moving obstacles. PEARL learns the task with only four static obstacles, and plans trajectories in densely populated environments with up to 900 obstacles. In real-time, at 10 Hz, the planner generates shorter trajectories of higher success rates compared to a traditional Gaussian Artificial Potential Field (APF) obstacle avoidance method, which requires extensive manual tuning [14]. In addition, trajectories generated by PEARL are shorter and have success rates comparable to state of the art collision avoidance algorithms, such as ORCA [23].

The contributions of this work are: 1) a solution for planning high-dimensional, preference-balancing, motion-based problems (PEARL), and 2) continuous RL solutions to dynamic obstacle avoidance. These contributions are achieved by Markov decision process

[1]Computer Science Department, University of New Mexico, USA {afaust@cs.unm.edu, lewispro@unm.edu, futuredragon@gmail.com, tapia@cs.unm.edu}

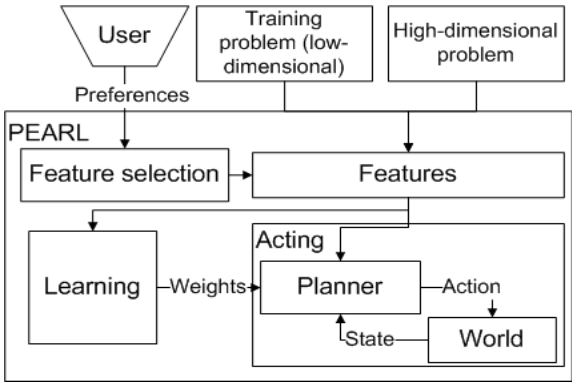


Fig. 1. PrEference Appraisal Reinforcement Learning (PEARL) framework for learning and executing PBT. The user-provided preferences are encoded into polymorphic features. The learning agent appraises preference priorities on a low-dimensional training problem. The planner takes an initial state of a high-dimensional problem and produces actions in a closed feedback loop.

formulation for PTBs, and obstacle avoidance tasks, formalizing the preference feature selection method, including preferences that increase over time, and appraising the preferences on small tasks with continuous action RL.

II. Related Work

Reinforcement Learning: Function approximation RL methods typically assume user-provided features [1]. Yet, selecting good features is very difficult because they map the entire robot's state subspace to a single point, and RL is very sensitive to feature selection [1]. Classically, two feature types are used in RL: discretization, and basis functions [26]. Discretization partitions the domain, scaling exponentially with the space dimensionality. Basis functions, such as kernels and radial basis function networks, offer more learning flexibility. These functions, however, can require manual parameter tuning, and the feature number increases MDP formulation for multi-robot systems and dynamic obstacle avoidance tasks exponentially with the space dimensionality [26]. PEARL proposes a feature selection method that solves a particular class of motion tasks for acceleration-controlled robots. PEARL features exploit task knowledge. Their number is invariant to the problem dimensionality, and the computation time scales polynomially with the state space dimension. In related work, Voronoi decomposition solves high dimensional manipulation problems by projecting the robot's configuration space onto a low dimensional task space [20]. The features we propose define a basis in the preference-task space as well. However, PEARL autonomously learns the relationship between the features. Another RL approach solves manipulation tasks with hard [21] and soft [13] constraints. Our tasks, however, do not have known constraints and bounds; they are set up as preferences to guide dynamically feasible trajectory generation.

In summary, PEARL takes task preferences as objectives, and generates features. It appraises the features to come up with weights. Another approach, Weighed Real-Time Heuristic Search (RTHS) [19], uses the weights along the search during planning. PEARL uses RL, which intrinsically includes weights to balance short and long term gains. In addition, PEARL learns weights between the features to find a good balance of priorities among opposing preferences.

Dynamic Obstacle Avoidance: Planning motions in dynamic environments is challenging because plans must be frequently adjusted due to moving obstacles. Sampling-based methods provide low cost solutions to high-dimensional planning problems in dynamic environments [10], [11], [17]. For example, lazy Probabilistic Roadmaps (PRM) accommodate moving obstacles by rechecking edge validity [10], [11], while [17] pre-building a Rapidly-exploring Random Tree (RRT) that is modified whenever obstacle changes are detected. Artificial Potential Fields (APF) provide low-cost solutions to dynamic environments by using only local information near the robot [14]. Nevertheless, several parameters impact the performance, such as the relative strength between the repulsive and attractive potentials and the size of the repulsive potential. ORCA is a state-of-the-art, multi-agent collision avoidance algorithm [23]. Each agent computes the velocity obstacle posed by other nearby agents and plans an action that reciprocally avoids collision. It can be modified for single agent, multiple dynamic obstacles planning [3].

III. Background

We consider robots as mechanical systems that can be moved using an external force, and model their motion with a special case of nonlinear systems, *a discrete-time control-affine system* [15]. Consider a robot with m degrees of freedom (DoF). If an acceleration $\boldsymbol{a}(n) \in \mathbb{R}^m$ is applied to the robot's center of mass at time-step n, the new position-velocity vector (state) $\boldsymbol{s}(n+1) \in \mathbb{R}^{2m}$ is,

$$D: \quad \boldsymbol{s}(n+1) = \boldsymbol{f}(\boldsymbol{s}(n)) + \boldsymbol{g}(\boldsymbol{s}(n))\boldsymbol{a}(n), \qquad (1)$$

for some functions $\boldsymbol{f}, \boldsymbol{g}$. A *Markov decision process* (MDP), a tuple (S, A, D, R) with states $S \subset \mathbb{R}^{2m}$ and actions $A \subset \mathbb{R}^m$, that assigns immediate scalar rewards $R : S \to \mathbb{R}$ to states in S, formulates a task for the system (1) [1]. A solution to a MDP is a control policy $\pi : S \to A$ that maximizes cumulative discounted reward over the agent's lifetime, value, $V(\boldsymbol{s}(0)) = \sum_{i=0}^{\infty} \gamma^i R(\boldsymbol{s}(i))$, where $0 \leq \gamma \leq 1$ is the learning constant. *Approximate value iteration* (AVI) [4], finds a solution to a continuous state MDP by approximating state-value function V with a linear map

$$\hat{V}(\boldsymbol{s}) = \boldsymbol{\theta}^T \boldsymbol{F}(\boldsymbol{s}). \qquad (2)$$

AVI takes a feature vector $\boldsymbol{F}(\boldsymbol{s})$ and learns weights $\boldsymbol{\theta}$ between the features by sampling the state-space and observing the rewards. It iteratively updates $\boldsymbol{\theta}$ in an expectation-maximization manner.

After the parameter learning is completed, batch RL enters a planning phase. The planner takes the value function approximation (2), an initial condition, and generates a trajectory using the closed-loop control with a greedy policy with respect to the state-value approximation,

$$\pi^{\hat{V}}(\boldsymbol{s}) = \underset{\boldsymbol{a} \in A}{\operatorname{argmax}}\, \hat{V}(\boldsymbol{s}'); \tag{3}$$

where state $\boldsymbol{s}'$ is the result of applying action $\boldsymbol{a}$ to state $\boldsymbol{s}$. Action selection in continuous spaces, which calculates the greedy policy (3), is a multivariate optimization over an unknown function. Several sampling-based methods that approximate the policy efficiently exist, such as Hierarchical Optimistic Optimization applied to Trees (HOOT) [16]. HOOT uses hierarchical discretization to progressively narrow the search on the most promising areas of the input space, thus ensuring an arbitrarily small error [16]. In practice, HOOT works well for single-agent planning with value functions that have many small-scale maxima.

IV. Methods

Our aim is to solve tasks that can be described with a set of goals (attractors), and obstacles (repellents) for acceleration-controlled robots with unknown dynamics. We require that the solution, PEARL, is efficient and adaptive. By efficient, we mean that PEARL controls agents in real-time in fully continuous and physically unbounded spaces; by adaptive, we mean that a single learning can be applied to a number of tasks.

PEARL solves PBT in two phases, learning and acting (Figure 1), adhering to the batch RL paradigm. To start the learning phase, a user provides PEARL with basic information about the problem; the robot's DoFs, maximum accelerations, etc., and a set of objectives (preferences). The basic system information is encoded into a MDP as presented in Section IV-A. Meanwhile, given the preferences, which consist of task goals and obstacles, PEARL generates the features using the methods described in Section IV-B.

With the MDP and features setup, the learning phase uses one of the AVI-based RL algorithms on a simplified problem space to discover the relative weights between the preferences (preference appraisal). The value function is approximated with Equation (2) and is a linear map of preference-based features. Once the preference weights are learned, they are handed over to the planner, and the acting phase can begin.

The acting is a closed-loop feedback system, or a decision-making that can work online, or plan trajectories offline in simulation. The planner solves problems with larger state and action domains, because the features are valid in the larger domain and capture the important elements of the task, rather than the physical space. The features enable both the efficiency, by learning on small problems, and adaptation, by allowing the policy transfer to larger problems. It is the use of the polymorphic, automatically generated features that separates PEARL from standard batch RL, and creates a virtually tuning-free task learning and completion method.

A. MDP Setup

For the **General PEARL Formulation**, we assume the robot works in continuous state and action spaces, is controlled through acceleration applied to its center of mass, and has dynamics that are not explicitly known. Let $\boldsymbol{s}$, $\dot{\boldsymbol{s}}$, $\ddot{\boldsymbol{s}} \in \mathbb{R}^{d_r}$ be the robot's position, velocity, and acceleration, respectively. The MDP state space is $S = \mathbb{R}^{d_s}$, where $d_s = 2d_r$, where d_r is the robot's DoFs. The state $\boldsymbol{s} \in S$ is joint vector $\boldsymbol{s} = [\boldsymbol{s}, \dot{\boldsymbol{s}}]^T$, and action $\boldsymbol{a} \in A = \mathbb{R}^m$ is the joint acceleration vector, $\boldsymbol{a} = \ddot{\boldsymbol{s}}$. The state transition function that we assume is unknown, is a control-affine dynamical (1).

We assume for training purposes the presence of a simulator or dynamics samples for the robot. The reward R is set to one when the robot achieves the goal, and zero otherwise. The tuple (S, A, D, R) defines the MDP for the robot problem.

For the **Dynamic Obstacle Avoidance Task**, the MDP setup is the joint vector of robot position and velocity, $S = \mathbb{R}^4$. The action space is the acceleration on each axis with dimension $A = \mathbb{R}^2$.

B. Feature Selection

For the **General PEARL Formulation**, we define a PBT with n_o objectives, $\boldsymbol{o}_1, ..., \boldsymbol{o}_{n_o}$, and preferences with respect to the objectives. The objectives, points in positional or velocity space, $\boldsymbol{o}_i \in \mathbb{R}^{d_{ri}}$, $i = 1, .., n_o$, either attract or repel the one or more agents. We call preferences that attract the agent *distance-reducing*, whereas the preferences that repel it are *intensity-reducing*; both preference types have the goal of reducing their measure to an objective.

To learn PBT with n_o objectives, $\boldsymbol{o}_1, ..., \boldsymbol{o}_{n_o}$, we form a feature for each objective. Assuming the low dimensional task space and high-dimensional MDP space $n_o \ll d_s$, we consider *task-preference features*,

$$\boldsymbol{F}(\boldsymbol{s}, d_s) = [F_1(\boldsymbol{s}, d_s), ..., F_{n_o}(\boldsymbol{s}, d_s)]^T.$$

Parametrized with the state space dimensionality, d_s, the features map the state space S to the preference space, and, depending on the preference type, measure either the squared intensity or distance to the objective. Let $\boldsymbol{p}^{\boldsymbol{o}_i}(\boldsymbol{s})$ be a projection of the robot's state onto the minimal subspace that contains $\boldsymbol{o}_i$. For instance, when an objective $\boldsymbol{o}_i$ is a point in a positional space, $\boldsymbol{p}^{\boldsymbol{o}_i}(\boldsymbol{s})$ is the robot's position. Similarly, when $\boldsymbol{o}_i$ is a point in

a velocity space, $\boldsymbol{p}^{\boldsymbol{o}_i}(\boldsymbol{s})$ is the robot's velocity. Then, *distance-reducing features* are defined with

$$F_i(\boldsymbol{s}, d_s) = \|\boldsymbol{p}^{\boldsymbol{o}_i}(\boldsymbol{s}) - \boldsymbol{o}_i\|^2, \tag{4}$$

and *intensity-reducing features* are defined with

$$F_i(\boldsymbol{s}, d_s) = (1 + \|\boldsymbol{p}^{\boldsymbol{o}_i}(\boldsymbol{s}) - \boldsymbol{o}_i\|^2)^{-1}. \tag{5}$$

Algorithm 1 summarizes the feature selection procedure.

Algorithm 1 PEARL feature selection.

Input: $\boldsymbol{o}_1, ..., \boldsymbol{o}_{n_o}$ objectives, $pt_1, ..., pt_{n_o}$ preference types
Input: MDP (S, A, D, R),
Output: $\boldsymbol{F}(\boldsymbol{s}, d_s) = [F_1(\boldsymbol{s}, d_s), ..., F_n(\boldsymbol{s}, d_s)]^T$
1: **for** $i = 1, \ldots, n_o$ **do**
2: **if** $pt1$ is intensity **then**
3: $F_i(\boldsymbol{s}, d_s) = (1 + \|\boldsymbol{p}^{\boldsymbol{o}_i}(\boldsymbol{s}) - \boldsymbol{o}_i\|^2)^{-1}$ {inensity preference}
4: **else**
5: $F_i(\boldsymbol{s}, d_s) = \|\boldsymbol{p}^{\boldsymbol{o}_i}(\boldsymbol{s}) - \boldsymbol{o}_i\|^2$, {distance preference}
6: **end if**
7: **end for**
8: return $\boldsymbol{F}(\boldsymbol{s}, d_s)$

For the **Dynamic Obstacle Avoidance Task**, there are two natural preferences: 1) minimize the distance to the goal, and 2) maximize the distance from obstacles. The feature vector is then formulated as the combination of the two preferences: $\boldsymbol{F}(\boldsymbol{s}) = [F_1(\boldsymbol{s}) \;\; F_2(\boldsymbol{s})]^T$. Providing Algorithm 1 with the goal's and obstacle's coordinates, the features are $F_1(\boldsymbol{s}) = \|\boldsymbol{s} - \boldsymbol{G}\|^2$, and $F_2(\boldsymbol{s}) = (\beta + d^2)^{-1}$, where $\boldsymbol{G}$ is the goal's position, d is the minimum distance to the closest obstacle, and β is a constant empirically selected to be 0.01.

V. Analysis

Feature properties: Features selected in this manner have the following properties allowing PEARL to have the potential to learn on small problems and transfer the learning to larger problems:

- *Feature domain:* The features are Lipschitz continuous and defined for the entire state space, $F_i : \mathbb{R}^m \rightarrow \mathbb{R}$, $i = 1, .., d_s$, in contrast to tiling and radial basis functions [26], [1] that are active only on a section of the problem domain.
- *Projection to preference space:* Features project the state subspace into a point that measures the quality of the preferences. Thus, the state-value function approximation from (2) is an d_s-dimensional manifold in the preference space, and their number does not change as domain space change.
- *State space polymorphism:* Because they are based on the vector norm and projection, the features are polymorphic with respect to the domain dimensionality. Since the learning is more computationally intensive than the planning, we use lower-dimensional problems for training. The feature vector size is invariant to the number of agents, state space dimensionality, and physical space dimensions. If the agents operate in 2D space, the features consider only planar space. But, when the same agents are placed in a 3D environment, the feature set remains unchanged even though the 3D space is considered in feature calculations [6].
- *Polynomial computation time:* The feature computation time is polynomial in state space dimensionality.

Local minima analysis: For tasks with mixed objectives, such as moving obstacle avoidance, the agents follow preferences, but there are no formal completion guarantees. In fact, the value function (2) has potentially two maxima, one on each side of the obstacle.

Since a straight line is the shortest path between an agent an its attractor, we analyze the value function restricted to that line with varying obstacle distances. To simplify, without loss of generality, we rotate and scale the coordinate system, such that the attractor is in the origin, and the agent is on the $x-$axis. The two nearest obstacles lay on (1, d), and (1, $-d$). Let c be ratio between learned weights for the obstacle and the attractor feature. The value function after the affine transformation is $V_x(x) = V(x, 0) = x^2 + \frac{c}{(x-1)^2+d^2}$. By examining the first and second derivative $\dot{V}_x$, and $\ddot{V}_x$ as a function of the obstacle distance d in relation to c, we conclude that (1) all learned weights must be negative in order for the agent to approach the attractor, and (2) the value function either has a single maxima near an attractor (pink, and blue lines in Figure 2), has two maxima (blue and green line), or has an inflection point near the obstacles (red line). An empirical study evaluates the method. Inspection of the partial derivative $\frac{\partial V}{\partial y}$ at the minima points in Figure 2 reveals that these points are saddle points.

In summary, when the obstacles are far enough apart there is only a global maximum. As the obstacles come closer together, a new region of attraction forms on the other side of the obstacle. If the agent gets into the local maximum region of attraction, gradient-descend methods will trap it. Sampling-based greedy methods such as HOOT, however, might get the agent out of the region of attraction if it is sufficiently close to the boundary.

VI. Results

We now demonstrate PEARL for the Dynamic Obstacle Avoidance Task. PEARL was implemented in MATLAB and executed on an Intel i5-4200U with 4GB RAM.

Figure 3 illustrates the testing environment and task for the point-like holonomic robot to navigate from

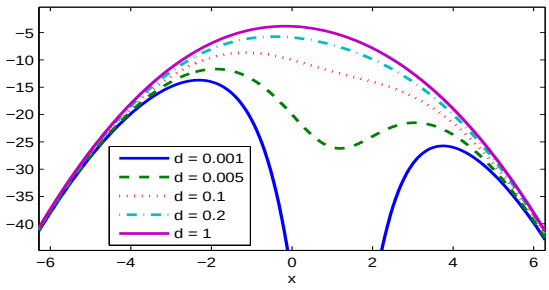


Fig. 2. Value function inflection points for $c = 100$.

the starting location to the goal without colliding with circular obstacles. The robot observes only the current position of the closest obstacle, and has no information about its heading. Each obstacle has a constant heading, but the speed is sampled stochastically every $\Delta T_{sample} = 1$ second.

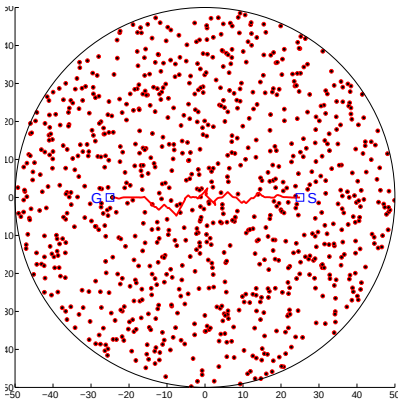


Fig. 3. Dynamic Obstacle Avoidance Task initial environment with 900 moving obstacles and an example path. The point-like robot must travel from the start (S) to the goal (G) while avoiding obstacles.

Learning Setup: We use 4 stationary obstacles placed at $[3\,\text{m}, 0\,\text{m}]$, $[0\,\text{m}, 3\,\text{m}]$, $[0\,\text{m}, -3\,\text{m}]$, $[-3\,\text{m}, 0\,\text{m}]$ to learn the weights between the two features. The goal is at the origin. The sampling space is inside a two-dimensional hypercube $[-5\,\text{m}, 5\,\text{m}]^2$. The robot has a maximum speed of $0.36\,\text{m/s}$, and a maximum acceleration of $3\,\text{m/s}^2$. We run AVI [4] with HOOT policy approximation [16], as the learning agent in PEARL for 300 iterations to learn the feature vector weights.

Learning Result: The resulting weights are $\boldsymbol{\theta} = [-0.23 \;\; -0.1696]^T$. All simulations are done at 10 Hz. The time to learn is 123 s.

Planning Environment Setup: The planning task environment is illustrated in Figure 3. The robot must travel from the start position $[25m, 0m]$ to the goal at $[-25m, 0m]$ under the same speed and acceleration constraints as used for learning. The environment has $N = \{300, 450, 600, 750, 900\}$ randomly placed moving obstacles following the constant heading with stochastic speed dynamics. The obstacles are circles with radius $r_{obs} = 0.5\,\text{m}$. The set of possible linear velocities is $\{0.1\,\text{m/s}, 0.2\,\text{m/s}, 0.5\,\text{m/s}, 0.7\,\text{m/s}\}$ with probabilities $\{0.3, 0.2, 0.3, 0.2\}$ respectively. The average speed of obstacles is identical to the maximum speed of the robot.

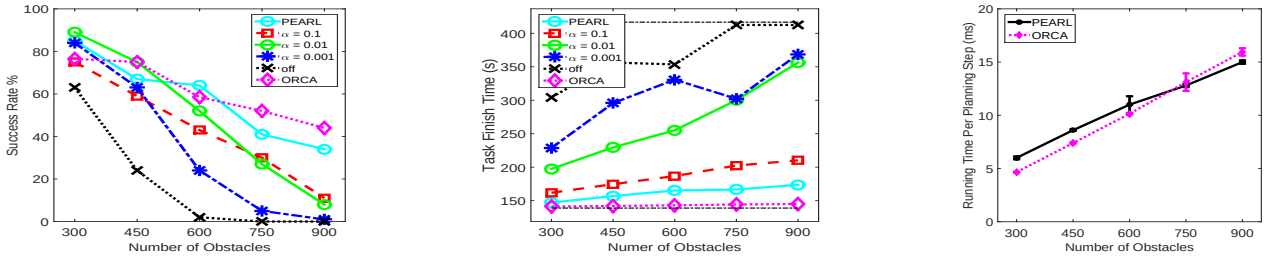
We maintain the constant density of moving obstacles by restricting the robot and moving obstacles to lie in a circle with radius 50 m. When an obstacle hits the boundary of the circle, it is transported to the antipodal position on the circle and continues evolving from this new position. The resulting density of moving obstacles is similar to [2].

We compared our method with two methods that only consider the current position (and velocity) of obstacles. The Gaussian APF method considers only the position of obstacles. It combines a linear attractive potential toward the goal and a repulsive potential from obstacles [14]. The obstacle potentials are Gaussians with $\sigma = 0.45\,\text{m}$ around obstacles, tuned empirically for this problem. The relative strength between the attractive and repulsive potential, α, has a significant impact on the success rate and needs to be manually tuned. Larger α represents a more goal-greedy robot behavior. We compared various values of α to our method. ORCA considers the position and velocity of obstacles [23]. A C++ implementation of ORCA (executed on the same machine as PEARL) was downloaded and modified for a single robot with multiple obstacles from [24]. We modified ORCA by disabling the velocity obstacle calculation and reciprocal collision avoidance for agents acting as moving obstacles. The agent acting as the robot still calculates velocity obstacles in order to avoid moving obstacles.

Planning Result: Figure 4a and 4b show that planning with PEARL has a higher probability of successfully avoiding obstacles, and reaches the goal in less time compared to the Gaussian APF method. The success rate and task finish time of the Gaussian method depends greatly on the parameter α. This parameter has to be tweaked manually or by optimization algorithms [22] after many planning trials. PEARL balances the features (similar to finding the optimal α) in the learning phase with a simplified scenario, and is able to transfer the weights to the online plan with comparable or better performance. ORCA and PEARL have a comparable success rate (less than 11% difference), indicating that it is a viable alternative method for dynamic obstacle avoidance. The running time per planning step for PEARL is comparable to ORCA, even though PEARL is implemented in MATLAB and ORCA is in C++. PEARL scales linearly with the number of obstacles and is capable of generating trajectories real-time (Figure 4c).

VII. Conclusion

This paper presents PEARL, a solution for high-dimensional preference-balancing motion problems,



(a) Success rate (b) Task Finish Time (c) Running time

Fig. 4. Trajectory characteristics for environments with varied complexity (number of obstacles). The agent planned with the same policy, learned with four static obstacles averaged over 100 trials. (a) Collision-free success rate of finishing the task, (b) Amount of time for the agent to reach the goal without collision, (c) and running time per planning step. The black line in (a) is the limiting case where the robot stops seeking the goal when any obstacle is within 2 m. PEARL has comparable success rate and running time per planning step.

that is efficient, adaptive, and controls the agent in real-time. The method uses features that work in continuous domains, scale polynomially with the problem's dimensionality, and are polymorphic with respect to the domain dimensionality. PEARL was demonstrated on a complex Dynamic Obstacle Avoidance Task where the agent has to progress toward the goal while avoiding collision with 900 stochastically moving obstacles. Our experiments show that PEARL outperforms the traditional Gaussian APF method and is comparable to ORCA, a state of art algorithm for the obstacle avoidance.

References

[1] L. Buşoniu, R. Babuška, B. De Schutter, and D. Ernst. *Reinforcement Learning and Dynamic Programming Using Function Approximators*. CRC Press, Boca Raton, Florida, 2010.

[2] H.-T. Chiang, N. Malone, K. Lesser, M. Oishi, and L. Tapia. Aggressive moving obstacle avoidance using a stochastic reachable set based potential fi eld. In *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, 2014.

[3] H.-T. Chiang, N. Rackley, and L. Tapia. Stochastic ensemble simulation motion planning in stochastic dynamic environments. In *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, 2015.

[4] D. Ernst, M. Glavic, P. Geurts, and L. Wehenkel. Approximate value iteration in the reinforcement learning context. application to electrical power system control. *International Journal of Emerging Electric Power Systems*, 3(1):1066.1–1066.37, 2005.

[5] A. Faust, I. Palunko, P. Cruz, R. Fierro, and L. Tapia. Learning swing-free trajectories for UAVs with a suspended load. In *IEEE International Conference on Robotics and Automation (ICRA), Karlsruhe, Germany*, pages 4887–4894, May 2013.

[6] A. Faust, I. Palunko, P. Cruz, R. Fierro, and L. Tapia. Automated aerial suspended cargo delivery through reinforcement learning. *Artificial Intelligence*, 2014.

[7] A. Faust, P. Ruymgaart, M. Salman, R. Fierro, and L. Tapia. Continuous action reinforcement learning for control-affine systems with unknown dynamics. *Automatica Sinica, IEEE/CAA Journal of*, 1(3):323–336, 2014.

[8] H. Hasselt. Reinforcement learning in continuous state and action spaces. In M. Wiering and M. Otterlo, editors, *Reinforcement Learning*, volume 12 of *Adaptation, Learning, and Optimization*, pages 207–251. Springer Berlin Heidelberg, 2012.

[9] T. Hester and P. Stone. TEXPLORE: real-time sample-efficient reinforcement learning for robots. *Machine Learning*, 90(3):385–429, 2013.

[10] L. Jaillet and T. Simeon. A PRM-based motion planner for dynamically changing environments. In *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, 2004.

[11] M. Kallman and M. Mataric. Motion planning using dynamic roadmaps. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 4399–4404, 2004.

[12] J. Kober, D. Bagnell, and J. Peters. Reinforcement learning in robotics: A survey. *International Journal of Robotics Research*, 32(11):1236–1272, 2013.

[13] T. Kunz and M. Stilman. Manipulation planning with soft task constraints. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1937–1942, October 2012.

[14] C.-P. Lam, C.-T. Chou, K.-H. Chiang, and L.-C. Fu. Human-centered robot navigation towards a harmoniously human–robot coexisting environment. *TR*, 27(1):99–112, 2011.

[15] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006.

[16] C. Mansley, A. Weinstein, and M. Littman. Sample-based planning for continuous action markov decision processes. In *Proc. of Int. Conference on Automated Planning and Scheduling*, 2011.

[17] M.Otte and E.Frazzoli. Rrt-x: Real-time motion planning/replanning for environments with unpredictable obstacles. In *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, 2014.

[18] K. Mülling, J. Kober, and J. Peters. A biomimetic approach to robot table tennis. *Adaptive Behavior*, 19(5):359–376, 2011.

[19] N. Rivera, J. A. Baier, and C. Hernandez. Weighted real-time heuristic search. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '13, pages 579–586, Richland, SC, 2013. International Foundation for Autonomous Agents and Multiagent Systems.

[20] A. C. Shkolnik and R. Tedrake. Path planning in 1000+ dimensions using a task-space voronoi bias. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 2061–2067. IEEE, may 2010.

[21] M. Stilman. Global manipulation planing in robot joint space with task constraints. *IEEE/RAS Transactions on Robotics*, 26(3):576–584, 2010.

[22] K. C. Tan and W. Ming-Liang. Evolutionary artificial potential fields and their application in real time robot path planning. In *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*, pages 256–263, 2000.

[23] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha. Reciprocal n-body collision avoidance. In *Robotics research*, pages 3–19. Springer, 2011.

[24] J. van den Berg, S. J. Guy, J. Snape, M. C. Lin, and D. Manocha. Rvo2 library: Reciprocal collision avoidance for real-time multi-agent simulation. `http://gamma.cs.unc.edu/RVO2/`.

[25] C. Wirth and J. Fürnkranz. Preference-based reinforcement learning:a preliminary survey. In *ECML/PKDD-13 Workshop on Reinforcement Learning from Generalized Feedback: Beyond Numeric Rewards*, Sep 2013.

[26] C. Wu. *Novel function approximation techniques for large-scale reinforcement learning*. PhD thesis, Northeastern University, Apr 2010.