2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
September 24–28, 2017, Vancouver, BC, Canada

Task-Based Behavior Generalization via Manifold Clustering

Rafael Garcia, Bruno C. da Silva, and João L. D. Comba

Abstract— **Machine learning algorithms can be expensive to deploy, in particular, those used in robotics applications that perform many variations of the same task. Solutions to one variation of a task may be found via Reinforcement Learning algorithms, and are typically modeled as a vector of N parameters encoding the robot's behavior policy. When N is large or executing robot trials is time-consuming, searching in the space of solutions becomes prohibitively expensive. In this paper, we introduce a method that allows robots to generalize behaviors by analyzing solutions to a small number of previously-trained related tasks. This allows for approximate policies for novel tasks to be rapidly estimated. We present a method that achieves this type of generalization by performing nonlinear regression directly on the *policy manifold* — i.e., the solution space spanned as we change the parameters describing tasks. Because tasks are typically described by few parameters, the corresponding policy manifold has few degrees of freedom, which leads to low-dimensional surfaces. We exploit this property to construct a function that maps task parameters to policy parameters (a *parameterized skill*). Our method uses manifold clustering techniques to deal with discontinuous manifolds, a challenging situation arising from physical obstacles or robot constraints. We evaluate our method on a set of robot manipulation tasks and show that it can efficiently estimate policies for novel tasks from a small number of training examples.**

I. INTRODUCTION

Machine learning algorithms typically require searching for solutions in high-dimensional spaces. This may be computationally costly if the search process requires collecting a large number of samples of the function being optimized. In certain domains, such as robotics, this problem becomes even more evident because each update to candidate solutions may require executing many trials in a physical robot—a process that can demand significant physical and time resources.

In this paper, we address the problem of how to optimize not a single optimization problem—called a *task*—but many variations of a task. We assume that our task is defined by a set of input parameters representing the specific variation to be solved. The solution to one task, then, consists of a set of parameters; in the case of robotics, the behavior that solves a particular task is often called a *policy* and encodes actions at each state in order to execute a given desired task.

In many cases, we do not want to solve a single task, but a large number of variations of that task. Consider, for instance, a robot whose objective is to change the position of objects in a room. In this example, any variation in object's target or initial position, or even in the properties of the object itself, results in a different task. Therefore, during its operation, the robot might need to accomplish an immeasurable amount of tasks. If the space of tasks is large, or infinite, it should be clear that it is not feasible for a robot to learn all possible task variations, and alternative strategies must be designed.

We aim at developing a method capable of finding solutions to any possible task in a given family of related tasks, without requiring the robot to solve it from scratch. Generally, different variations of a task may share similarities; consider, for example, the problem of moving a robot's actuator in some space packed with obstacles. One particular task in this setting could be represented by a pair of initial and final positions; a corresponding policy for solving a task can be represented as a high-dimensional vector of parameters encoding the physical movements that the robot must execute to complete the desired movement. In this case, even though there are infinite tasks that the robot could execute, it should be clear that policies/behaviors may change only slightly and smoothly in response to small changes in the parameters of a task. In particular, a surface of continuous tasks would be mapped to a surface of continuous policies.

Formally, the set of policies for solving a family of tasks lies in a low-dimensional manifold embedded in the high-dimensional policy space [1]. If the manifold's geometry is known in advance, one could directly calculate the policy for any desired input task. Unfortunately, this is not the case in real robotics applications, since analytically modeling the manifold's geometry requires a priori knowledge about the behaviors that optimize arbitrary tasks in a given task family—which may require exact knowledge about the environment and robot's dynamics. We present an approach using Manifold Clustering (MC) techniques to estimate the geometry of the policy manifold using a small number of samples. This can be exploited to construct effective nonlinear regression models that directly predict points *on* the manifold—the surface which encodes policy parameters for novels, yet-to-be-learned tasks—thereby avoiding the costly process of running an optimization process for each task.

Estimating the geometry of nonlinear manifolds, and performing nonlinear regression on them, is hard because one often does not know how complex the policy manifold is—in particular, at which locations the policy space is discontinuous, and how nonlinear different regions in that space are. If these factors are poorly estimated, the quality of the resulting regressions may be arbitrarily low. In the previously-mentioned example, manifold discontinuities and varying degrees of nonlinearity may result from physical constraints to the robot, which may cause tasks with similar parameters to require qualitatively different policies. Concretely, a regressor would need to model a function whose outcomes can vary in non-smooth ways in response

All authors are with the Instituto de Informática at the Federal University of Rio Grande do Sul (UFRGS), Brazil. {rgarcia,bsilva,comba}@inf.ufrgs.br.

to smooth changes to its input. Naively constructing a single regression model, mapping task parameters to policy parameters, thus covering the entire policy manifold, could lead to poorly performing predicted policies, since the regressor would not model discontinuities and different degrees of nonlinearity in different input regions.

We introduce a method that can efficiently estimate, via a sampling process, the geometry of the policy manifold—including discontinuities and regions with different complexities—to estimate reasonable solutions or policies for novel tasks. We present an adaptive framework capable of representing a complex, possibly disjoint manifold, as a set of smaller and simpler locally-continuous charts, thereby making it easier to construct accurate regressors from task parameters to high-dimensional policies that solve those tasks. This framework can then be applied to rapidly estimate the solution to new robotics optimization problems without having to solve each possible problem or task from scratch.

II. SETTING

Consider a robot that needs to solve a large number of tasks drawn from an arbitrary task distribution. We assume that each task is modeled as a Markov Decision Process (MDP) [2], and that each single task must be executed by a robotic agent in a way that maximizes some measure of reward. Since the agent may need to execute many tasks, it should aim at maximizing not the expected reward at individual tasks, but the expected reward over all the tasks that may be drawn from the distribution of possible MDPs. We consider the case where MDPs have similar dynamics and reward functions so that they can be considered variations of a single task with a same general objective in common. Formally, the agent's goal in this paper is to maximize

$$\int P(\tau)J(\pi_\theta, \tau)d\tau, \tag{1}$$

where τ is a $|T|$-dimensional vector drawn from a continuous space T representing tasks that the robot may need to perform by learning an appropriate policy. A policy π_θ is parameterized by a vector $\theta \in \mathrm{R}^N$. When a given policy π_θ is used to solve a task τ during an episode of length t_f, it returns an expected reward $J(\pi, \tau) = E\{\sum_{t=0}^{t_f} r_t|\pi, \tau\}$ measuring how efficient π_θ in solving τ. Finally, some tasks may have a higher probability of being performed than others. To represent the relative importance of finding good solutions for these tasks, we add a term $P(\tau)$ representing the probability of task τ occurring. Let a *parameterized skill* be a function mapping tasks to corresponding policies:

$$\Theta : T \rightarrow \mathrm{R}^N. \tag{2}$$

Note that when tackling a distribution of tasks, the exact policy parameters to be used depends on the current task being solved by the agent; these can be obtained by Θ, so that the agent uses policy $\pi_{\Theta(\tau)}$ when solving task τ. The goal of the agent, in this case, is to estimate a parameterized skill function Θ that maximizes $\int P(\tau)J(\pi_{\Theta(\tau)}, \tau)d\tau$.

A. Assumptions and Goals

The main goal of our approach is to design a method capable of estimating the solution to a large number of tasks drawn from a distribution $P(\tau)$. Assume we have access to a set K of pairs $\{\tau, \theta_\tau\}$, where τ is a $|T|$-dimensional vector of task parameters sampled from $P(\tau)$ and θ_τ is the policy vector that maximizes the expected reward for task τ. We wish to use K to create a parameterized skill that approximately maximizes the objective described in Section II. In what follows, we assume, for simplicity, that $P(\tau)$ in a uniform distribution; in practical cases where different tasks have different probabilities of occurring, one needs only to change the sampling procedure used to construct K; see Section III for more details.

The number of task variations one can draw from P usually is infinite. However, related tasks usually require using similar behaviors. In most cases, smooth variations in task parameters generate smooth variations in the corresponding policy vector. Therefore, one can assume that the set of policies that optimize our objective function (Section II) does not cover homogeneously the N-dimensional space they are inserted. Instead, they lie in a lower-dimensional manifold formed by the behavior of the variations presented by the policies. The assumptions that smooth changes to tasks parameters result in smooth changes to their corresponding policies is a theoretical possibility, but a property often observed in real-life robotics applications. Consider again the example from Section I. If we modify the robot's goal position by a few centimeters but keeping the same starting position and the same held object, the policy suitable to solve this new task variation should be just slightly different from the original task policy. Similarly, if we repeat this process continuously, we will get a set of policies that form a low-dimensional manifold embedded in N, whose shape is given by the policies behavior, or how they change in order to solve similar tasks. However, as previously stated, in specific points this may not occur, as obstacles or physical restrictions may force very different solutions for close tasks. In other words, a continuous surface in the task manifold is not mapped to a continuous surface in the policy manifold at that point. This implies the existence of a manifold formed by multiple continuous charts. One of our goals is to identify these charts in order to effectively learn each of them independently.

III. BEHAVIOR GENERALIZATION VIA MC

Let a training set $D = \{\tau_i, \theta_i\}_{i=1,\ldots,|K|}$ be comprised of a few policies that solve task-based behaviors. The sampling method used to generate D depends on the probability density function P from where the tasks are drawn. Non-uniform distribution requires sampling that prioritizes tasks variations with a greater probability of being performed. Regression methods are commonly used to generalize the behavior of a function associated with an input set. There is a wide range of regressions techniques, each with its pros and cons. However, a regression method is not always suitable to correctly estimate policies manifolds. As explained in

Section I, these manifolds may have a non-uniform surface—with some regions more complex than others—and discontinuities may exist over its domain. Our approach deals with the problem by breaking the manifold surface into smaller regions, called charts, with continuous domain and somewhat uniform complexity. To achieve this, one must identify these charts from the sampled data. Traditional classifiers and clustering techniques are not always suitable for this job, as they generally consider spatial distance as the affinity between samples. This measure is not effective as a single continuous chart might have samples that are far apart in terms of Euclidian distances, but near in the actual policy space/manifold being considered. Similarly, two data points close in the high-dimensional space may belong to different charts. Therefore, alternative clustering methods able to deal with those characteristics are needed.

To solve this problem, we use the MC [10] technique called *Sparse Manifold Clustering and Embedding* (SMCE) proposed by E. Elhamifar and R. Vidal in [3] to identify how many continuous charts exist in the manifold formed by the policies θ. In brief, SMCE identifies charts in the manifold by applying spectral clustering to a $|K| \times |K|$ matrix M containing the affinity between each pair of samples. These affinities are defined as the solutions to sparse l_1-optimization problems that aim to identify, for each sample policy θ_i, a few samples in the neighborhood of θ_i that can linearly reconstruct θ_i. More specifically, for each sample θ_i, a set V_i is constructed with the $|V_i|$ nearest neighbors of θ_i. This set is then used as input to the optimization problem specified in Equation 3, where Q_i is a diagonal matrix with weights that penalize points that are far apart from θ_i, and whose diagonal values are calculated as shown in Equation 4. The set of weights c_i is used as affinities to fill in the i-th row of the matrix M—matrix entries associated with points that do not belong to V_i are set to have affinity 0, including M_{ii}.

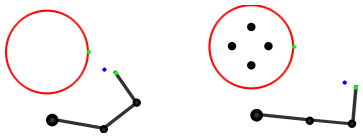
$$\min \lambda||Q_i c_i||_1 + \frac{1}{2}|||V_i - \theta_i|c_i||_2^2 \text{ subject to } \mathbf{1}^T c_i = 1 \quad (3)$$

$$Q_{i\{j,j\}} = \frac{||V_{ij} - \theta_i||_2}{\sum_{t \in V_i} ||t - \theta_i||_2} \quad (4)$$

The matrix M encodes in each row i a set of coefficients that allows a point θ_i in the manifold to be reconstructed in terms of other points in the manifold. If M_i^j is small, the point θ_j does not contribute significantly to determining θ_i. This usually implies that these points θ_i and θ_j are far apart on the manifold surface or are not in the same continuous chart. Given M, we can recover an arbitrary number S of charts by applying spectral clustering to M. There is a trade-off between higher and smaller values for S. As the number of clusters increases, the complexity of each chart becomes smaller but the number of required regressions increases—and the number of samples available for each of them decreases. However, methods such as the Elbow Method result in very efficient lower-bound estimations for S.

Each row i of M can be interpreted as the axes of an N-dimensional space, where the axes of this space measure the similarity of θ_i with respect to each of the other N tasks in the training set. Clustering the points constructed based on M (i.e., its rows) results in groups of policies θ that can be reconstructed similarly. Policies lying in the same clusters are close to each other in the manifold surface and therefore clustered in the same chart. In other words, policies that solve similar tasks τ tend to require similar behaviors θ. When this does not happens and similar tasks are placed in different clusters, that indicates that there is a discontinuity in that region of the manifold surface.

Fig. 1. Reaching a given location in two different workspaces.



Lastly, for each chart, nonlinear regression is applied to estimate each of the N attributes of the policy vector for a previously unlearned task. In this work, we implemented the regression using Gaussian Processes since they are capable of modeling surfaces with non-uniform complexity.

IV. Results

We evaluated our method in a simulated robotic arm tasked with reaching arbitrary positions within a 2D workspace with obstacles placed at unknown locations. The robotic arm is composed of three actuated links. Tasks in this problem correspond to the locations within the workspace where a target may be placed, and policies correspond to functions specifying the behavior of the robot—in particular, the sequence of joint angles necessary for it to safely move from a starting position to a given target position. Note that obstacles may restrict the movements of the robot, which creates discontinuities in the policy space, since nearby tasks (positions in the workspace) may require qualitatively different movements or policies. Figure 1 depicts two examples of the simulated environments we consider, both with and without obstacles. The possible goals positions within this domain lie over a circumference (shown in red) that surrounds the arm; black circles in Figure 1 are the obstacles.

We encode tasks via a unidimensional parameter in the interval of 0 to 360, corresponding to the angular position of the target over the circumference. The space of behaviors in this problem is highly redundant, since many sequences of angles may exist that allow the robot to move safely to a given target position. To address this problem, we define the execution performance of a given policy via a cost function that takes into account its accuracy (the distance between the end-effector of the robot and the target at the end of the movement) and a regularization term that penalizes longer movements, thereby helping the system to identify unique solutions more easily and breaking ties based on movement complexity. Policies are represented using the framework of Dynamic Movement Primitives, or DMPs [4]. DMPs are parametric policies that allow for a robot to compactly

represent complex trajectories in arbitrary spaces way. They are parameterized nonlinear differential equations whose time evolution encodes smooth kinematic control policies that drive the movement of a robot, as described below:

$$kv = K(g - x) - Qv + (g - x_0)f \quad (5)$$

$$kx = v, \quad (6)$$

where x is the current state within the trajectory being modeled, v is the velocity, x_0 and g define the starting and goal states of the trajectory, K is a spring constant, and Q is a damping term; finally, k is a temporal scaling factor determining the execution speed of the trajectory. In Equation 5, f is a linearly parameterized non-linear function whose parameters can be learned by the agent, thereby allowing it to adjust the DMP's trajectory so that it encodes appropriate task-specific behaviors. This function is defined according to Equation 7, where $\psi_i(s)$ are Gaussian basis functions that dependent on the current phase (or time) s of the system, and that have adjustable parameters or weights w_i. The phase variable s decreases monotonically from 1 to 0 along the execution of the movement and is computed by integrating $ks = -\alpha s$, where α is a pre-defined constant.

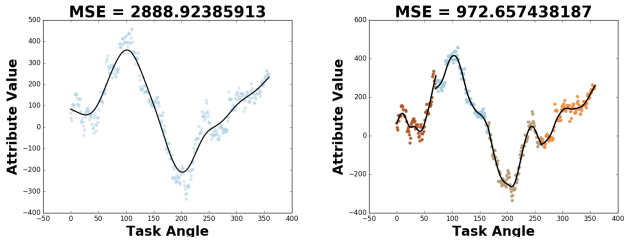
$$f(s) = \frac{\sum_i w_i \psi_i(s)}{\sum_i \psi_i(s)} \quad (7)$$

$$\psi_i(s) = \exp(-h_i(s - c_i)^2). \quad (8)$$

By using the DMP framework, arbitrary trajectories (e.g., in the space of joint angles) can be generated by integrating the set of differential equations presented in Eq 5. In this paper we use DMPs to encode sequences of Euclidean coordinates of the robot's end-effector; the coordinates generated by a DMP are then processed by an inverse kinematics algorithm and the resulting angles fed to a PID controller. In the experiments below, we parameterized a DMP (in particular, its non-linear function f) via a 6-dimensional parameter vector θ; modifying it results in different candidate trajectories that the arm may choose to follow to achieve a given goal position.

In this paper, we pose the problem of identifying the best policy for solving a given task as a Markov Decision Process (MDP). The rewards resulting from the execution of a policy are given by the cost function mentioned before—one that trades off accuracy and movement complexity. Finding a policy that maximizes the expected reward of a policy can be done via any Reinforcement Learning (RL) algorithm. Note, however, that standard RL algorithms optimize a single policy, suitable for solving a single task. In our case, by contrast, we wish to rapidly identify policies that are appropriate for any tasks within a possibly infinite task space; executing an RL policy optimization process from scratch for any novel task would be infeasible. By applying our method, we can learn a parameterized skill function Θ that directly predicts policies θ_τ for arbitrary novel tasks τ. The first step in this process involves constructing the training set D (Section III). To construct D, we need to sample tasks along the task domain and search for policies that solve these tasks. Generally, any policy search algorithm can be used to calculate these policies. In this paper, we use the Covariance Matrix Adaptation Evolution Strategy algorithm (CMA-ES [9]) to find policies that solve individual tasks.

Fig. 2. One selected dimension of the policy manifold spanned by 360 task samples, collected from an obstacle-free environment. Single regression (left); multiple regression models from automatically manifold clustering into four charts (right).



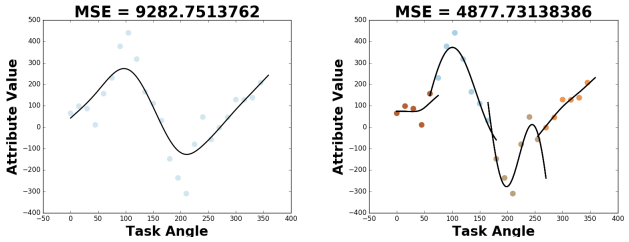
A. Experiment 1: Obstacle-Free Environment

Figure 2 shows 360 points from one selected dimension of the 6-D manifold formed by a set P containing sample solutions in an obstacles-free setting. This was numerically computed by enumerating tasks τ up to a minimum resolution of 1 degree and using the CMA-ES method to manually compute an optimal policy for that task. Since this manifold is globally continuous, smooth changes to the task (angle) result in smooth changes to the policy. As the manifold has a single chart, it can be estimated by a single regression. However, as shown in Figure 2 (left), the prediction may bring poor results due to the high complexity of the manifold surface. We solve this issue by clustering the manifold in charts with smoother surfaces, that can be easily predicted. Figure 2 (right) shows the regressions applied after clustering the data into four charts, which results in a lower mean square error than the former approach. If we cluster the manifold into more charts, we can improve prediction performance. However, the number of required regressions increases significantly and results in fewer points being available to each regressor, thus undermining the results. The number of charts was defined by the Elbow Method behavior.

Searching for policies is an expensive task. Therefore, we must ensure that only a few number of policies are required to estimate the manifold surface. For this purpose, we considered 24 evenly spaced samples from P, which we call P'. Figure 3 shows one selected dimension of the samples in P', with both approaches applied. We observe that although the small number of samples naturally make it difficult for the regressors to recover all the features, our method still has an overall better prediction for novel tasks, thus resulting in a smaller average MSE. Figure 3 shows the regression behavior for both approaches. When no clustering is applied (left), the prediction when P' is used as training set is significantly worse than the predictions obtained when P is used. On the other hand, the predictions have higher quality when MC is applied. Figure 8 (left) depicts the increase of the MSE as the number of samples decreases for both approach plus an alternative method called CrKr (see IV-C).

In some problems, a small perturbation applied to the

Fig. 3. One selected dimension of the policy manifold spanned by 24 task samples, collected from an obstacle-free environment. Single regression (left); multiple regression models from automatically manifold clustering into four charts (right).



policy used to solve a task may change the quality of the solution. In these cases, even small regression errors may significantly degrade the quality of the executed policy. To investigate this further, we validated the quality of our solutions by executing the predicted policies in a simulation. Figure 4 compares the efficiency of the training done with and without MC in the simulation, using 24 previously trained samples. On top of Figure 4, we show the execution error when no clustering is applied, and only one estimator per attribute is used, while at the bottom, we show the execution error when clustering is performed, and four charts are clustered. Both the average and the standard deviation of the execution error decreases when clustering is applied.

B. Experiment 2: Environment with Obstacles

In this experiment, we evaluate how our approach performs in an environment with obstacles (Figure 1 (right)). We generate a sample data set Q with 360 6-dimensional points corresponding to the policies associated with each corresponding task (Figure 5). We depict in Figure 5 one selected dimension of the policy manifold. The manifold shown here is composed of four charts (number found after applying the Elbow Method), resulting from movement restrictions caused by the presence of obstacles in the environment. When no

Fig. 4. Comparing execution error of predicted policies: the x-axis shows the task domain (angle of the goal position) while the y-axis shows the execution error of the policies predicted by the Gaussian Process (distance in centimeters between the end effector and the target position). The average and the standard deviation of the execution error are also highlighted. The top image shows the execution error for a single estimator and the bottom image shows the execution error using four estimators retrieved by the MC.

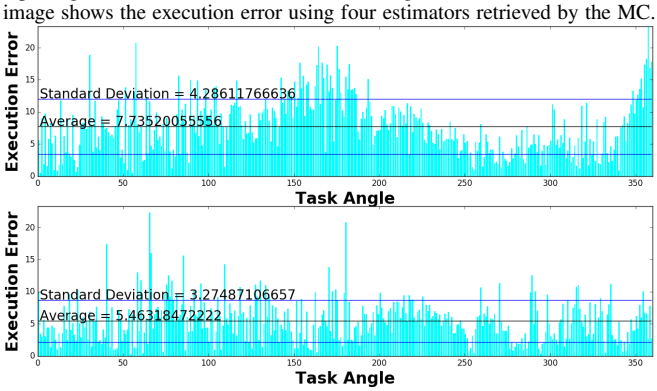


Fig. 5. One selected dimension of the policy manifold spanned by 360 task samples, collected from an environment with obstacles. Single regression (left); multiple regression models from automatically manifold clustering into four charts (right).

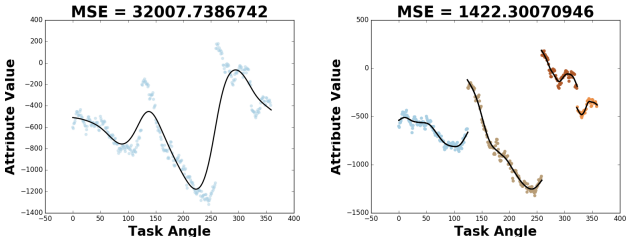
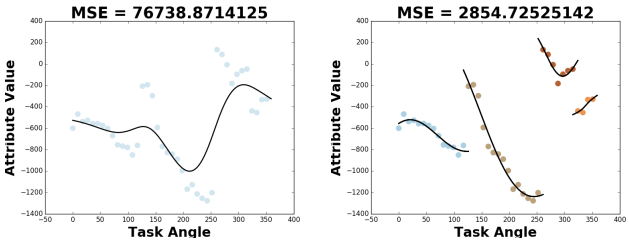


Fig. 6. One selected dimension of the policy manifold spanned by 24 task samples, collected from an environment with obstacles. Single regression (left); multiple regression models from automatically manifold clustering into four charts (right).



clustering is used (left) the results have poor predictions, as a single regression tries to fit a continuous curve over many discontinuous regions. However, by using MC we identify those regions and constructing an independent estimator on each chart and thereby achieving better predictions.

The presence of discontinuities makes it hard for a single estimation method to efficiently learn the manifold surface, as charts often bias the estimation of its neighbors, especially at boundaries. However, this becomes even worse when only a few samples are used. In this case, the manifold surface cannot be correctly estimated at all. Figure 6 (left) shows these effects. Our approach can handle this situation, as shown in Figure 6 (right). As we are able to detect the charts boundaries and apply independent estimators to each of them, we predict better policies. Figure 7 compares the execution error of predicted policies when running in the simulation with and without MC. The advantages of MC are even more salient here because of the stronger discontinuities in the datasets. Figure 8 (right) shows how the MC approach is able to predict solutions for novel tasks with small errors even in discontinuous environments, while other methods struggle even when the number of samples is higher.

C. Comparison with CrKr

We compared our method against another existing approach, called *Cost-regularized Kernel Regression (CrKr)* [8]. This method aims at generalizing behaviors for motor primitives based on adapting meta-parameters via a reward-weighted regression. It is a non-parametric method that efficiently models complex surfaces. However, it does not offer a suitable solution when the manifold has discontinuities along its domain. Figure 8 shows the average

Fig. 7. Comparing performances in an environment with obstacles using a single estimator (top) and when the charts are previously clustered (bottom)

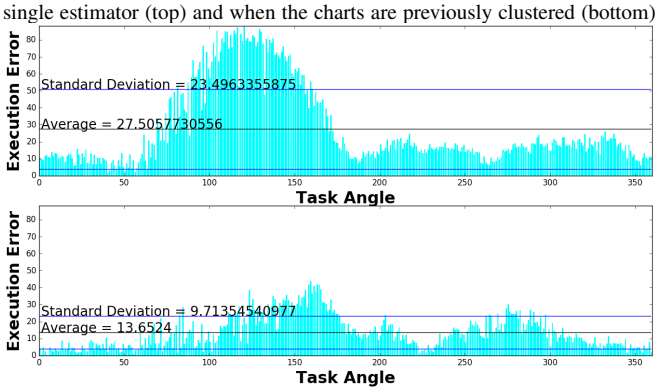
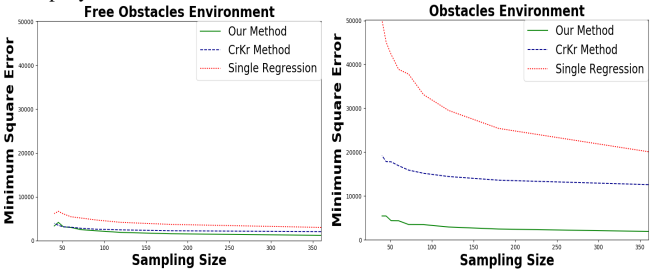


Fig. 8. Comparing predictions achieved by our method, the CrKr [8] and when a single regression is used. Both graphics use the same scale to exemplify the effectiveness of the method in discontinuous manifolds.



minimum square error as the number of samples increases. Our method performs better in both environments (with and without obstacles). The biggest improvement happens when the manifold has strong discontinuities. In this case, *CrKr* is not able to successfully predict the surface even with a large number of samples. On another hand, our method has stable effectiveness even for small sampling sizes (as low as 50)—seven times smaller than the complete data set.

V. Related Work

Several strategies have been proposed recently to handle parameterized skill learning. Soni and Singh [5] propose the construction of hierarchies of skills allowing terminating criteria to be adapted and thus being able to deal with closely related tasks. However, their approach is not able to find policies to solve new tasks, as they do not directly build a parameterized skill. Liu and Stone [6] propose an algorithm mapping a source to a target task through Bayes networks. As their method requires previous knowledge of the tasks dynamics, it may not be able to handle some applications. Hausknecht and Stone [7] manually select a relevant task parameter to be exhaustively tested, in order to speed up the process of finding optimal policies and estimate a skill. Kober, Wilhelm, Oztop, and Peters [8] propose *CrKr*, a nonparametric regression—discussed in the previous section—able to generalize behaviors by estimating the mean and variance values of the DMP meta-parameters. Although it can effectively handle complex manifolds, it is not able to deal with discontinuities in the manifold. Mullig, Kober, Kroemer and Peters [12] propose a method that selects policy samples through kinesthetic imitation from humans behaviors and generalizes them via a mixture of movement primitives selected by a Gating Network. However, it requires policies to be linear in their parameters. Deisenroth, Englert, Peters, and Fox [13] generalize a single learned policy to multiple tasks using a parameterization of both task and system state. Da Silva, Konidaris, and Barto [1] propose a framework to create parameterized skills similar to ours, but that uses ISOMAP and classification algorithms to recognize discontinuous charts in the policy manifold.

VI. Conclusion

We introduced a framework to learn parameterized skills that generalize robot behaviors to novel tasks using a small number of samples. We address issues arising from performing regression analysis directly in policy space, which might be discontinuous. We handle this problem by using manifold clustering techniques over policy samples and modeling the geometry of each disjoint chart via a separate Gaussian Process. Our method significantly reduces the required number of samples, thus also reducing the amount of computation needed to learn the parameterized skill. The quality of our predictions is similar to the one achieved by learning policies from scratch but without the agent to do so. This occurs since predictions are more robust to eventual discontinuities that may arise due to obstacles and movement constraints.

References

[1] da Silva, B.C.; Konidaris, G.; Barto, A.G., Learning Parameterized Skills, in Proceedings of the 29th International Conference on Machine Learning (ICML 2012). Scotland, 2012.

[2] Bellman, R., A Markovian Decision Process, in Indiana University Mathematics Journal 6 No. 4 (1957), 679-684.

[3] Elhamifar, E.; Vidal, R., Sparse Manifold Clustering and Embedding, in Neural Information Processing and Systems, 2011.

[4] Schaal, S.; Peters, J.; Nakanishi, J.; Ijspeert, A., Learning movement primitives, in Proceedings of the Eleventh International Symposium on Robotics Research. Springer, 2004.

[5] Soni, V.; Singh, S., Reinforcement learning of hierarchical skills on the Sony Aibo robot, in Proceedings of the Fifth International Conference on Development and Learning, 2006.

[6] Liu, Y.; Stone, P., Value-function-based transfer for reinforcement learning using structure mapping, in Proceedings to the Twenty-First National Conference on Artificial Intelligence, pp. 415-420, 2006.

[7] Hausknecht, M.; Stone, P, Learning powerful kicks on the Aibo ERS-7: The quest for a striker, in RoboCup-2010: Robot Soccer World Cup XIV, volume 6556 of Lecture Notes in Artificial Intelligence, pp. 25465. Springer Verlag, 2011.

[8] Kober, J.; Wilhelm, A.; Oztop, E.; Peters, J., Reinforcement learning to adjust parametrized motor primitives to new situations, Autonomous Robots, 33(4):361379, 2012. ISSN 0929-5593.

[9] Auger, A.; Hansen, N., Tutorial cma-es: Evolution strategies and covariance matrix adaptation, in Proceedings of GECCO'12 (Conference Companion on Genetic and Evolutionary Computation), 2012.

[10] Izenman, A. J., Introduction to manifold learning, in Wiley Interdisciplinary Reviews: Computational Statistics, John Wiley and Sons, Inc., v. 4, n. 5, p. 439-446, 2012.

[11] Rasmussen, C.; Williams, C., Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning) MIT Press, 2005.

[12] Mullig, K.; Kober, J.; Kroemer, O.; Peters, J., Learning to select and generalize striking movements in robot table tennis, in International Journal of Robotics Research Vol 32, Issue 3, 2013

[13] Deisenroth, M. P.; Englert, P.; Peters, J.; Fox D., Multi-task policy search for robotics, in 2014 IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, 2014, pp. 3876-3881.