In. Proc. IEEE/RSJ International Conference on Intelligent Robotics and Systems (IROS), Taiwan (2010).

Using Dimensionality Reduction to Exploit Constraints in Reinforcement Learning

Sebastian Bitzer Matthew Howard Sethu Vijayakumar

***Abstract*— Reinforcement learning in the high-dimensional, continuous spaces typical in robotics, remains a challenging problem. To overcome this challenge, a popular approach has been to use demonstrations to find an appropriate initialisation of the policy in an attempt to reduce the number of iterations needed to find a solution. Here, we present an alternative way to incorporate prior knowledge from demonstrations of individual postures into learning, by extracting the inherent problem structure to find an efficient state representation. In particular, we use probabilistic, nonlinear dimensionality reduction to capture latent constraints present in the data. By learning policies in the learnt latent space, we are able to solve the planning problem in a reduced space that automatically satisfies task constraints. As shown in our experiments, this reduces the exploration needed and greatly accelerates the learning. We demonstrate our approach for learning a bi-manual reaching task on the 19-DOF KHR-1HV humanoid.**

I. Introduction

The application of reinforcement learning (RL) to continuous, high-dimensional systems such as anthropomorphic robots (Fig. 8) remains a challenging problem. While a large variety of RL algorithms exist for solving complex planning problems [1], typically the scalability of these is limited to applications involving small, discrete worlds. Continuous state spaces necessitate discretisation, or the use of function approximators, but both are affected by the curse of dimensionality, that states that the resources needed to solve a learning problem scale exponentially with the dimensionality of the state space.

In this context, recent attention in the robotics community has focused on this issue of scaling RL to higher-dimensional problems. For example, in a programming by demonstration framework, demonstrated trajectories can be used to initialise a parametrised policy [2]. Because such an initial policy is assumed to be close to the optimal policy, only a limited number of policy updates may be needed to find an acceptable solution. Hierarchical RL [3] is a more general approach in which the RL problem is broken down into a hierarchy of sub-problems, solutions of which are combined to solve the high-level problem. This divide and conquer approach is intuitively plausible, but the difficulty is then shifted towards selection and learning of the hierarchy. Despite recent advances [4], problems remain, in particular with large state spaces. Abstractions [5] have been suggested as a general term describing a mapping of state space to a more compact, abstract space which benefits learning. The simplest abstraction, for example, just selects a subset of the state dimensions, but any transformation of the state space is possible. If an insight into the control problem exists a priori, an abstraction can be chosen by hand [6], but ideally we would like to learn suitable abstractions from experience.

In this paper, we investigate the suitability of dimensionality reduction (DR) as a method for automatically determining abstractions for RL from demonstrations and the conditions for the success of this approach. While the idea of using DR to aid RL has recently been explored by Morimoto et al. [7], to find a low dimensional state representation that preserves the reward structure, unfortunately, using their approach only uses a DR technique (i.e., Kernel DR) which, in many problems, is not sufficient to represent the state space faithfully (see Sec. IV). In contrast, our contribution shows that the GPLVM, as a non-linear DR method based on Gaussian Processes (GPs) [8], can produce much more faithful state representations for simplifying the learning problem. Using such an approach, we show the feasibility of RL for very high-dimensional robotic systems, even when no initialisation of the policy is available. We illustrate our approach for learning a bi-manual reaching task on the 19-DOF KHR-1HV humanoid robot.

II. Programming by Demonstration Framework

The idea of our approach is to use data acquired from expert demonstrations to extract a non-linear manifold capturing the inherent structure of the data. The latter is then used as a reduced representation of the system state that can be exploited to improve the efficiency of RL. A schematic of the approach is illustrated in Fig. 1.

A. Extracting the Latent Space for Reinforcement Learning

In our framework, *kinesthetic demonstration* (in which the robot's movements are manually guided and recorded) is used to generate a set of postures that are deemed useful for the task by the demonstrator. Using kinesthetic demonstrations in this way has several benefits, for example, (i) it ensures that all demonstrated postures are feasible for the robot, (ii) the demonstrator can directly see that task constraints are satisfied within the demonstrations and (iii) it avoids correspondence issues that may arise due to differences in embodiment between the demonstrator and imitator (since the demonstrations are already performed on the robotic plant).

One of the downsides of kinesthetic demonstration is that it becomes increasingly difficult to demonstrate movements

S. Bitzer (s.bitzer@ed.ac.uk), M. Howard and S. Vijayakumar are with the School of Informatics, University of Edinburgh, Informatics Forum, 10 Crichton Street, EH8 9AB, UK. This work was funded under the EC FP6 project SENSOPAC (www.sensopac.org).

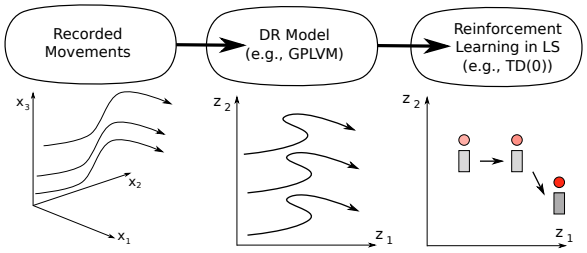


Fig. 1. Schematic of our approach. Given a set of demonstrated movements, DR is used to find a low-dimensional manifold on which the demonstrations lie (the latent space). The space defined by this manifold can then be used as the state-space representation within reinforcement learning.

(i.e., continuous trajectories) with increasing number of degrees of freedom of the robotic plant. To counter this, in our framework we use *discrete demonstrations*, where desired postures are demonstrated individually[1] (similar to 'keyframing' in animation). Specifically, in our framework, demonstrations are recorded by first moving all robot joints to the desired posture, recording the joint angles, and then repeating the procedure for the next one. In this way, even a single person can, with ease, provide demonstrations to a high-DOF humanoid to generate full-body movements.

Using the demonstrations, we then apply nonlinear DR techniques to extract a manifold that captures the latent structure of the data. In effect, here DR acts as a nonlinear interpolator that allows us to generate continuous movements from a discrete set of samples. At the same time, it provides a state representation which makes RL feasible even when the dimensionality of the original state space is very high.

We assume that the demonstrated postures fulfil constraints, when they are necessary for the achievement of the task (see example below). By introducing this invariance into the demonstrations, this latent structure (i.e., the constraint manifold) can be incorporated into the state space model learnt by the DR[2]. This, in turn, benefits RL by restricting exploration to parts of the space in which (in the eyes of the demonstrator) a feasible solution to the task exists.

B. Example: Constrained Bi-manual Manipulation

As a simple example of the above, consider a bi-manual manipulation task in which we want to move an object with two hands from one place to another (see Fig. 2). If the full state of the two arms is defined by the positions of the shoulder and elbow joints, then the total dimensionality of the system is four. However, for the movement to succeed, the condition that the two hands must remain a fixed distance apart must be fulfilled throughout the movement (see Fig. 2, left), otherwise the object will be dropped. In effect, this constrains the possible movements that can be used to solve the task: one degree of freedom is eliminated by constraining the distance between the two hands, and a second is eliminated if we do not allow rotations of the object. In other words, for this problem, any successful movement (fulfilling these task constraints) must lie on a 2-D manifold embedded in the full 4-D state space.

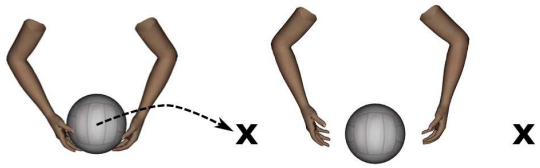


Fig. 2. In order to move the ball to the target (**x**), the movement must be constrained so that the hands remain a fixed distance apart (left). If the constraint is broken, the ball is dropped (right).

Now, if we can find an an appropriate representation of this manifold, then we can exploit it by restricting RL to only explore in the space where the constraints are satisfied. In some cases, this could be derived from an expert analysis of the task (here, involving derivation of kinematics of the plant and the constraints) resulting in an analytical model of the manifold. However, for the non-expert user, this greatly increases the complexity of finding the appropriate representation. Instead, in our framework, we propose to *learn the manifold by demonstration*. That is, we rely on the demonstrator to provide appropriate example postures that (i) satisfy the task constraints (here, postures in which the hands are in the right position to grasp the object), (ii) have sufficient coverage to make a reasonable approximation of the underlying manifold, and (iii) define a space in which a feasible solution to the task (here, a path to the target) exists. In other words, by taking appropriate care in selecting example postures, a non-expert demonstrator can use our framework to automatically learn a state representation that captures structural elements of the task (in this example, an implicit model of the constraints) without the need to define them formally by hand.

While this is a simple example, similar arguments also apply to more complicated situations, in particular for natural movements with many degrees of freedom such as that of humans [10] or humanoid robots [11] subject to more complex environmental or task constraints [12]. For systems such as these, using DR is even more appealing since formal definition of the task structure is much harder as the system dimensionality increases. In the next section we turn to the implementation details of the proposed framework.

III. Method

In this section, we describe the design choices made for implementing our programming by demonstration framework. As mentioned in the preceding sections, we assume that a non-expert demonstrator provides a number of kinesthetic demonstrations of key postures for a given task. These come in the form of vectors of joint angles $\mathbf{q}_n$, from which we wish to learn a nonlinear manifold that captures salient elements of the task. Having done this, we can then apply RL to find the optimal policy within the space defined by the learnt manifold, in order to find a feasible solution to the task.

[1]Please note that, if demonstrations of continuous movements are available, our approach can still be applied to find a compact state representation for RL. In this case, the sample density will simply be higher and, potentially, the sequential structure of the demonstrations could be exploited.

[2]This has interesting parallels with the idea of looking for *generalised coordinates* in analytical dynamics (e.g., see [9]) where, under a holonomic constraint (i.e., an *equality constraint*), it is possible to find a coordinate system in which the constraint is automatically satisfied. This greatly simplifies the problem of solving the equations of motion of the system.

A. Dimensionality Reduction

A number of DR techniques are available for extracting the latent structure from our demonstrations. In our setting, we require a method that (i) is able to represent manifolds that are potentially non-linear in the robot's joint space, and (ii) gives good generalisation with relatively little data (to minimise the number of demonstrations required for learning). Furthermore, in order to incorporate our DR model into RL, the DR technique must provide both a generative mapping and its inverse (i.e., generative mapping from latent to joint space, and the inverse mapping back).

By far the simplest and most popular approach to DR is to use principal components analysis (PCA) [13] which defines a linear mapping between low-dimensional latent space and high-dimensional data space based on eigenanalysis of the data covariance. It is robust and computationally efficient, but, as a linear technique, is not adequate for our purposes (as we show in Sec. IV). Our method of choice is the Gaussian Process Latent Variable Model (GPLVM) [14], a nonlinear extension to PCA based on Gaussian processes (GPs). In the next section we briefly review the formulation of the GPLVM employed in our experiments.

B. The Gaussian Process Latent Variable Model

The GPLVM defines a generative, probabilistic model of the data which uses GPs to map latent variables $\mathbf{z} \in \mathbb{R}^d$ to observed variables $\mathbf{x} \in \mathbb{R}^D$. Each data dimension $j \in 1, \ldots, D$ has its own GP, but all GPs share the same covariance parameters. In particular, the data likelihood of the model is defined as

$$P(\mathbf{X}|\mathbf{Z}, \boldsymbol{\theta}) \propto \prod_{j=1}^{D} \exp\left[\mathbf{x}_j^\top \mathbf{K}^{-1} \mathbf{x}_j\right] \tag{1}$$

where $\mathbf{X} \in \mathbb{R}^{N \times D}$ is the data matrix containing N data points, $\mathbf{x}_j$ is a column of this matrix, $\mathbf{Z} \in \mathbb{R}^{N \times d}$ is the matrix of latent points, $\boldsymbol{\theta}$ is a vector of covariance parameters and $\mathbf{K}$ is the covariance matrix of the GPs which depends on $\mathbf{Z}$ and $\boldsymbol{\theta}$. We use the standard squared exponential covariance function with added white noise defined as

$$K_{mn} = k(\mathbf{z}_m, \mathbf{z}_n) = \theta_1 \exp\left(-\frac{\|(\mathbf{z}_m - \mathbf{z}_n)\|^2}{2\theta_2}\right) + \delta_{mn}\theta_3 \tag{2}$$

where $\mathbf{z}_m$ and $\mathbf{z}_n$ are latent points, θ_1 controls the amplitude of the modelled function, θ_2 controls its smoothness and θ_3 is the variance of the Gaussian noise around the data. This formulation of the GPLVM can be derived from the dual formulation of probabilistic PCA which integrates out the parameters of the PCA model as shown in [14].

The positions of the latent points, $\mathbf{Z}$, and the covariance parameters, $\boldsymbol{\theta}$, are found simultaneously by minimising the negative GP data log-likelihood

$$\{\mathbf{Z}, \boldsymbol{\theta}\} = \arg\min_{\mathbf{Z}, \boldsymbol{\theta}} \; -\log \mathrm{P}(\mathbf{X}|\mathbf{Z}, \boldsymbol{\theta}) \tag{3}$$

using gradient descent. We initialise $\mathbf{Z}$ with points found by applying PCA as suggested in [14]. Because the inverse of $\mathbf{K}$ needs to be computed, each gradient step has a complexity of $O(N^3)$ which means that the learning gets expensive with increasing number of data points. In our setting, however, where the number of data points (as discrete demonstrations) this is not a significant problem.

More important in our setting is the speed of prediction since this is required at every time step during the RL (see Sec. III-C). In the GPLVM this is standard GP prediction which has $O(N)$ complexity, because $\mathbf{K}^{-1}$ is fixed after learning and can be pre-computed. Prediction for a single data point $\mathbf{z}^*$ returns a Gaussian distribution with mean μ_j and variance σ_j^2 in data dimension j

$$\mu_j = \mathbf{k}^{*\top} \mathbf{K}^{-1} \mathbf{x}_j \qquad \sigma_j^2 = k^* - \mathbf{k}^{*\top} \mathbf{K}^{-1} \mathbf{k}^* \tag{4}$$

where $k^* = k(\mathbf{z}^*, \mathbf{z}^*)$ is the covariance function evaluated at that point and $\mathbf{k}^* = [k(\mathbf{z}^*, \mathbf{z}_1), \ldots, k(\mathbf{z}^*, \mathbf{z}_N)]^\top$. The returned variance (equal in all data dimensions) gives a measure for the confidence in the prediction, usually indicating the quality of generalisation away from the data. In our setting, this relationship can be exploited in the RL to prevent the expensive evaluation of states for which the predictive variance indicates that the generated posture is unlikely to adhere to the task structure anyway (see Sec. IV-C).

The GPLVM only learns the mapping from latent to data space, but does not provide the mapping back. Many DR methods have the same problem and various out-of-sample extensions have been suggested. In our experience, the most accurate of these for the GPLVM projects a test point $\mathbf{x}^*$ into latent space by maximising its probability under the predictive distribution $N(\mathbf{x}^*|\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$ by varying the corresponding $\mathbf{z}^*$ with gradient descent only on $\boldsymbol{\mu}$ (4). Unfortunately this iterative procedure is comparatively slow, even though only a few iterations are needed, if initialised with the nearest data point. Consequently, we fit another set of GPs for the data-to-latent mapping after the GPLVM has been learnt. The resulting mapping has good accuracy in high confidence regions of the latent space and is efficient to compute.

C. Reinforcement Learning

A number of RL techniques are available for planning and optimising movements based on exploration of the environment. For the experiments in this paper, we restrict ourselves to the popular class of methods known as temporal difference learning (TD(0)) [1] since these perform robustly without the need for careful initialisation of parameters. In the following we briefly describe TD(0) learning for approximating the value function, as used in our framework.

D. TD(0) V-Learning

The general goal of learning is to find a policy $\boldsymbol{\pi}(\mathbf{u}|\mathbf{x})$ that maximises

$$V^{\boldsymbol{\pi}}(\mathbf{x}) = E_{\boldsymbol{\pi}}\left\{\sum_{t=0}^{\infty} \gamma^t r_t | \mathbf{x}_0 = \mathbf{x}\right\} \tag{5}$$

under dynamics $\mathbf{x}_{t+1} = \mathbf{x}_t + \delta t\, \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t)$. Here, $\mathbf{x} \in \mathbb{R}^n$ denotes the (continuous) state, $\mathbf{u} \in \mathbb{R}^d$ the action and δt is the time step. $V^{\boldsymbol{\pi}}(\mathbf{x})$ is the expected return accumulated by the agent when following the policy $\boldsymbol{\pi}$ starting from state $\mathbf{x}_0$, γ is a discount factor and r_t denotes the instantaneous

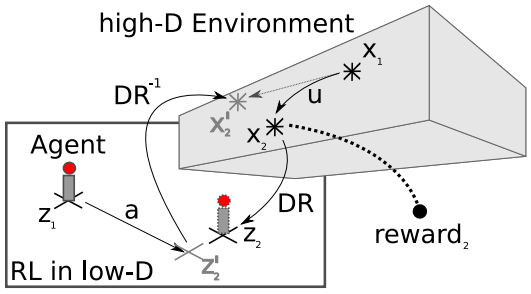


Fig. 3. Exploiting the latent dimensionality of demonstrations for RL.

reward collected at time t. $V^{\boldsymbol{\pi}}(\mathbf{x})$ can also be identified as the value function of $\boldsymbol{\pi}$.

TD(0) methods update the estimate of the value function or Q-function based on the one-step temporal difference (a.k.a. the Bellman error) [1]. In our experiments, we used the variant of TD-learning that uses a function approximator of the form

$$\hat{V}(\mathbf{x}) = \mathbf{w}^T \mathbf{b}(\mathbf{x}) \tag{6}$$

to learn (5). Here, $\mathbf{w} \in \mathbb{R}^M$ is a vector of weights, and $\mathbf{b}(\mathbf{x}) \in \mathbb{R}^M$ is a vector of fixed basis functions. For the latter we used normalised radial basis functions (RBFs) $b_i(\mathbf{x}) = \frac{K(\mathbf{x}-\mathbf{c}_i)}{\sum_{j=1}^{M} K(\mathbf{x}-\mathbf{c}_j)}$ calculated from squared exponential kernels $K(\cdot)$ around M pre-determined centres $\mathbf{c}_i,\ i = 1 \ldots M$.

During episodes, the value function is learnt online according to

$$\hat{V}(\mathbf{x}_{t+1}) = \hat{V}(\mathbf{x}_t) + \alpha \delta_t \tag{7}$$

where α is the learning rate and δ_t is the temporal difference

$$\delta_t = r_t + \gamma \hat{V}(\mathbf{x}_{t+1}) - \hat{V}(\mathbf{x}_t). \tag{8}$$

For our parametric model (6), this means we apply the update

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha \delta_t \mathbf{b}(\mathbf{x}_t). \tag{9}$$

Finally, using the approximated $\hat{V}(\mathbf{x})$, actions are selected according to a soft-max policy to provide directed exploration during episodes. Specifically, actions are drawn from a discrete set of $|U|$ continuous actions $\mathbf{u}_i \in \mathbb{R}^d$, $i = \{1, \ldots |U|\}$ according to the Boltzmann distribution

$$p(\mathbf{u}_i|\mathbf{x}) = \frac{\exp(\beta\, \hat{Q}(\mathbf{x}, \mathbf{u}_i))}{\sum_{j=0}^{|U|} \beta\, \hat{Q}(\mathbf{x}, \mathbf{u}_j)} \tag{10}$$

where β controls the rate of exploration and $\hat{Q}(\mathbf{x}, \mathbf{u}_i)$ is the state-action value for action $\mathbf{u}_i$, calculated using one-step look-ahead on the learnt value function, i.e.,

$$\hat{Q}(\mathbf{x}, \mathbf{u}_i) = \hat{V}(\mathbf{x} + \delta t\, \mathbf{f}(\mathbf{x}, \mathbf{u}_i)). \tag{11}$$

For further details of the implementation see [15].

E. Incorporating the Latent Space State Representation

For including the representation learnt with DR into our RL framework, we replace the high-dimensional state $\mathbf{x}$ with its DR representation $\mathbf{z}$, and modify the state update equations accordingly. Fig. 3 illustrates this for a single RL step.

Starting from a state in latent space, $\mathbf{z}_1$, RL selects and executes action $\mathbf{a}$ according to its current policy, leading to a new latent space state $\mathbf{z}'_2$. The latter is then used to generate a target in the environment $\mathbf{x}'_2$ by mapping through the generative GPLVM model, which can be reached, for example, using a simple PD controller. In general, (due to tracking errors, noise, etc.) we will not exactly reach $\mathbf{x}'_2$ but instead a slightly different state $\mathbf{x}_2$ which we must estimate (e.g., by taking a sensor reading). It is at this point that we receive a reward (i.e., based on the true environmental state). Finally, we return to latent space via the inverse mapping (out-of-sample GP) to estimate the new reduced state $\mathbf{z}_2$, which is then used to select the next action. Note that, due to the non-linearity of the DR mapping, the same action executed in different locations of latent space may correspond to different movements in the environment. This, however, is not a problem, if a suitably flexible local policy is chosen (i.e., one that selects actions based solely on the current state). Also note that, since the RL is restricted to the smaller latent space, it may not be possible to find globally optimal (or even feasible sub-optimal) solutions if they do not lie on this manifold. In practice, however, this is easily rectified by the demonstrator by, for example, adjusting the demonstrated poses and re-learning the DR model.

IV. Experiments

In this section, we report experiments exploring the performance of learning for systems of varying complexity and size. First, in order to illustrate the concepts involved, we apply our method to a simulated 4-DOF toy system with linear state dynamics. We then test the scalability of the method to a more complex, non-linear system and, finally, we illustrate the use of our approach for learning on the 19-DOF KHR-1HV humanoid robot (Fig. 8).

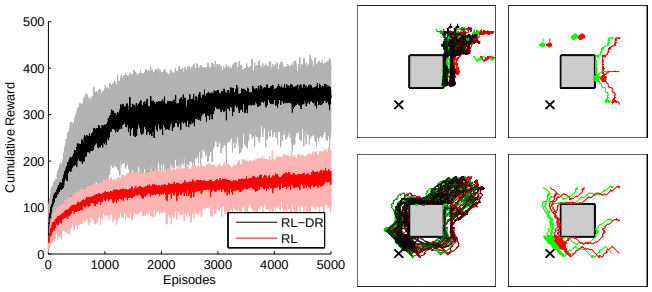
All our experiments are based on the intuitive example of carrying an object to a target using a bi-manual strategy, similar to the example described in Sec. II-B. The task of the learner is to find a movement that brings the hands to a target $\mathbf{x}^*$ without dropping the object. For this, the learner is rewarded according to

$$R(\mathbf{x}) = \exp\left\{-\theta \|\mathbf{x} - \mathbf{x}^*\|^2\right\} \tag{12}$$

where θ is a scaling parameter. Under (12), the learner receives very little reward over most of the space, but this rapidly increases as the hands approach $\mathbf{x}^*$. To increase the difficulty of this problem, we also placed an obstacle in the environment obstructing the path to the target. Accordingly, the learner was penalised if the hands (i) hit the obstacle or, (ii) hit the boundaries of the state space. In both cases, a fixed penalty $R_0 = -1$ was added to the reward and the episode terminated. Equal penalisation occurred at any time the object was dropped. As described in Sec. II-B, one of the keys to success in this task is, therefore, to maintain the hands on either side of the object throughout movement. Formally, this can be expressed as a set of constraints on the hands of the robot. Note, however, that this information is *not explicitly available* to the learner and therefore must be learnt either (i) from experience (i.e., exploration), or (ii) from the examples given to the learner as demonstrations.

A. Bi-manual Reaching in End-effector Space

Here, we formulate the bi-manual reaching problem in end-effector space and assume that the full state of the system can



(a) (b) (c)

Fig. 4. (a) Cumulative reward over episodes for the two-hand problem when learning in the full 4D state space (red) and the reduced 2D space (black). The mean±s.d. over 25 trials are shown. (b) Left (green) and right (red) hand trajectories generated at equal intervals throughout 5000 episodes of training in 4D (top) and 2D (bottom). Darker colours indicate later phases of learning. (c) 5 test trajectories sampled from different starting points after 5000 episodes of training with the two approaches. The grey area indicates the location of the obstacle and the target is marked with an 'x'.

be described by the horizontal-planar coordinates of the two hands, $\mathbf{x} \in \mathbb{R}^4$. State transitions followed linear dynamics, i.e., $\mathbf{x}_{t+1} = \mathbf{x}_t + \delta t\, \mathbf{u}_t$ with time step $\delta t = 0.1$ and we placed a square obstacle ($0.5 \times 0.5\, m$) in the environment (see Fig. 4). The target was at $\mathbf{x}^* = (0, 0, 0.1, 0)^T$ and we set $\theta = 0.75$ in the reward function (12). We compared two approaches to learning in this setting, namely (i) standard RL, whereby we used TD(0) to learn the optimal policy in the full state space $\mathbf{x}$ and (ii) the proposed approach, whereby a reduced state representation is learnt from demonstrations, prior to applying RL.

For the direct RL approach, the set up was as follows. The learner was given a set U of actions $\mathbf{u}_i \in \mathbb{R}^4$ allowing it to move the hands either independently or simultaneously, in the four orthogonal directions in $\mathbf{x}$. A Gaussian RBF network (6), with centres placed on a $20 \times 20 \times 20 \times 20$ grid was used to approximate the value function. A soft-max policy (10) was used to select actions where, to encourage exploration, we set $\beta = 10.0$. As parameters to the RL, we chose learning rate $\alpha = 0.9$ and discount factor $\gamma = 0.95$.

For the proposed approach, we randomly sampled 200 points across the space, which fulfilled the constraint on the distance between the hands. These were used to train the GPLVM to learn a reduced, 2D state representation, within which we applied TD(0). For the latter, all RL parameters were identical to those used for direct RL, with the exception that (i) the number of RBFs used in the value function approximation was scaled down to a 2D (as opposed to 4D) grid of 20×20 bases in latent space, and (ii) the learner's action set was reduced to that of movement in the four orthogonal directions in latent space. Please note that, for both approaches, if the global optimal policy is found, the learner can reach the target in the same number of steps, with the same reward.

Training was conducted for 5000 episodes, with each episode lasting 500 steps ($50\, s$). Start states were drawn from a Gaussian distribution $\mathcal{N}(\mathbf{x}_0, 0.1)$ around the point $\mathbf{x}_0 = (1, 1, 1.1, 1)^T$. To evaluate learning performance, the experiment was repeated for 25 trials and the reward accumulated in each episode of learning was recorded. The results are shown in Fig. 4.

	Bi-manual TS	Bi-manual JS
PCA	0.05 ± 0.01	3.25 ± 0.41
GPLVM	0.24 ± 0.18	0.80 ± 0.70
PCA	100.00 ± 0.00	4.92 ± 0.81
GPLVM	94.50 ± 5.61	61.03 ± 6.16

TABLE I

TOP: RECONSTRUCTION ERROR (RMSE$\times 10^2$) ON 1000 RANDOM POINTS IN END-EFFECTOR SPACE. BOTTOM: PERCENTAGE OF POINTS IN LATENT SPACE THAT FULFIL CONSTRAINTS (CF. FIG. 5). SHOWN ARE MEAN±S.D. OVER 20 TRIALS.

As can be seen, initially, the average reward accumulated by the two learners increases rapidly. However, when learning in the full 4D space, beyond the first 500 episodes the average reward starts to level out and the progress of learning is slow. In contrast, learning in the reduced dimensional space proceeds much quicker, with convergence already after approximately 3000 episodes. The reason for the performance difference becomes clear when looking at the trajectories generated during training. In Fig. 4(b) we show examples of trajectories generated at regular intervals during training with the two approaches. Clearly, due to the higher dimensionality, learning in the full 4D state space requires far more exploration to cover the same proportion of space. The trajectories generated during training also appear to be shorter than those generated with the DR representation, despite both having to avoid the same obstacle and boundaries. The difference is that the trajectories generated in the reduced space *automatically satisfy the constraint on the hands*. This means that exploration is focused only on the reduced part of the space in which possible solutions lie, and exploration of actions that lead to the object being dropped is avoided. As a result, the learner using DR rapidly learns a policy that allows it to satisfy the constraints and reach the goal from a larger range of the state space (compare example trajectories in Fig. 4(c)).

It should be noted that, in this simple example there in fact exists a simple linear transformation between the 2D constrained space and the 4D space of the hand positions. Consequently, linear PCA also gives good results (and even outperforms the GPLVM as shown in Table I) in this experiment. In our next experiment, however, the nonlinear relationship between the spaces introduced through the kinematics necessitates the use of nonlinear DR techniques.

B. Bi-manual Reaching in Joint space

In our second experiment we investigated a similar problem to that described in the previous section, with the difference that the task must be achieved by controlling 2 planar, 2-link arms. The constraints of the problem are identical (i.e., to keep the end-effectors keep a fixed distance apart) and full state space is still 4D, but instead of end-effector positions, the state is described by the joint angles of the robot. Due to the kinematics of the arms, non-linearities are introduced into the problem which cannot be handled by linear DR. To illustrate this point, we first compare the GPLVM with PCA and evaluate their ability to represent the constraint.

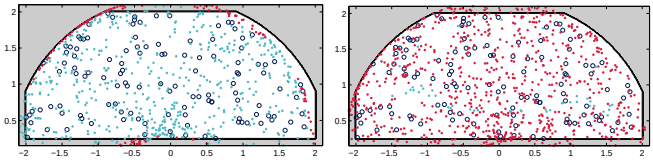


Fig. 5. Evaluating DR for the planar 2-link arms: Do points generated from latent space keep a distance of 0.1 between the two end-effectors? Light blue dots: end-effector position of left arm (right arm not shown) for which the distance between end-effectors lies within 0.005 of 0.1, red dots: distance differs by more than 0.005 from 0.1, circles: demonstrations used for DR. Top: GPLVM, bottom: PCA. In both figures the configuration of the arms is plotted for one example point.

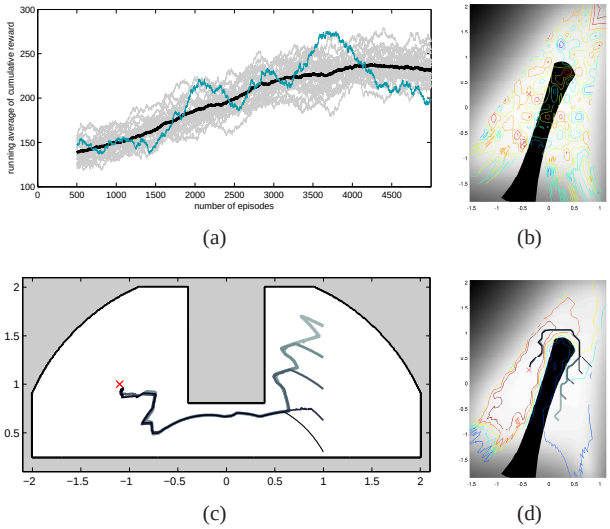
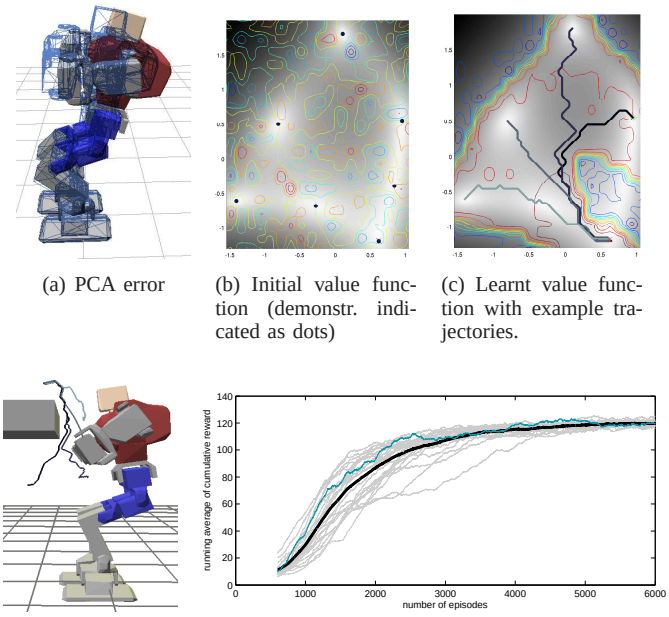


Fig. 6. RL results for the planar 2-link arms. (a) Rewards during learning, thick black line: average over the plotted trials, blue line: trial for which results in (b)-(d) are plotted. (b) and (d) Value function before and after learning. (c) Example trajectories in end-effector space (left hand only), red cross: goal. Shading in (b) and (c) visualises predictive variance of GPLVM generative mapping (white means low variance, high confidence). Black object is latent space representation of the obstacle after mapping through out-of-sample GP of GPLVM.

Fig. 5 shows a visualisation of our simulation in which several data sets are plotted. For clarity all data points shown are from the left hand of the robot only. The circles depict 123 randomly sampled data points which fulfil the constraints. We executed DR on their joint space representation, drew uniform samples from the resulting latent space and then mapped these to joint angles of the robot using the generative DR mapping. The dots are the corresponding hand positions as computed with the forward kinematics of the robot, colour-coded as to whether they fulfil the constraint within a small error margin. The results clearly show that PCA (Fig. 5, right) can only correctly represent the constraint in a very small region of end-effector space while the GPLVM (Fig. 5, left) covers almost the complete work space. Table I further documents this result.

Having established that the constraints are correctly represented by the GPLVM we ran RL in its latent space. We used the above setup with the following changes: we set the width of the Gaussian reward to $\theta = 0.35$, learning rate to $\alpha = 0.8$, discount factor to $\gamma = 0.99$, time step to $\delta t = 0.05$, soft-max policy to $\beta = 20$, extended the action set to also include diagonal actions and ran the learning for 5000 episodes with



(a) PCA error (b) Initial value function (demonstr. indicated as dots) (c) Learnt value function with example trajectories.

(d) Example traj. in end-effector space. (e) Rewards during learning, thick black line: average over the plotted trials, blue line: trial for which results (b)-(d) are plotted.

Fig. 7. (a) KHR-1HV in a demonstrated pose reconstructed by the GPLVM (wire frame) and PCA (solid). There is no visual difference between the GPLVM pose and the original demonstration. (b)-(e): RL results for the KHR-1HV. Shading in (b) and (c) visualises predictive variance of GPLVM generative mapping (white means low variance, high confidence).

1000 steps each (or the episode was stopped prematurely under the conditions given above). We again introduced an obstacle which, this time, only allowed successful trajectories to pass through a 'corridor' in end-effector space (Fig. 6(c)). Start states were drawn uniformly across latent space. The results are presented in Fig. 6.

Fig. 6(a) shows running averages over a window of 500 episodes of the cumulative reward per episode for 25 runs of RL (trials). We plot running averages, because the random start states mean that the cumulative reward per episode is highly variable. The accumulated reward clearly increases with learning. For the trial highlighted as the blue line, we present the initial and learnt value functions in latent space in Figures 6(b) and 6(d), respectively. As demonstrated by the sample trajectories in Fig. 6(d) the learnt policy successfully solves the task, leading trajectories around the obstacle into the goal. Fig. 6(c) depicts the resulting trajectories in end-effector space. For clarity we only plot the trajectories of the left hand, but right hand trajectories follow with the desired distance of $0.1\,m$ behind the left hand.

C. Full-Body Humanoid Reaching

In our final experiment, we demonstrate the complete approach on the 19 DOF KHR-1HV humanoid (Fig. 8). Similar to the preceding experiments, we investigate a bi-manual task, this time to lift an object while avoiding obstacles. Instead of devising an inverse kinematics for this task by hand, we demonstrated individual poses of 2 alternative ways of lifting an object (7 poses in total). Of the 19 DOF, 10 were major contributors to the changes in posture, the remaining 9

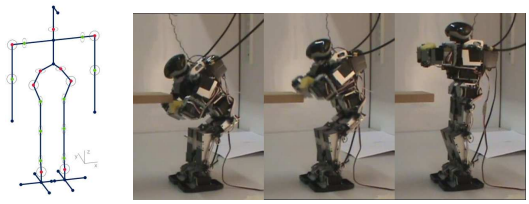


Fig. 8. Left: Kinematic model showing the 19 DOF of the robot.Orientation of circles indicates axis of rotation of rotation of the joints; x: vertical ellipse, y: circle, z: horizontal ellipse. Right: Video frames of successful, obstacle avoiding trajectory executed on KHR-1HV after RL.

(marked in red in Fig. 8(b)) changed by less than 10 degrees across different postures. The realised postures all lay on a central y-z-plane of the robot, i.e., the hands of the robot did not move sidewards (subject to noise originating from the manual demonstrations). Therefore, the space of related movements was inherently 2D which motivated the use of 2D latent spaces for DR. Compared to the previous examples, PCA already performed remarkably well in reconstructing and interpolating the demonstrated postures. The remaining inaccuracies, however, meant that the robot leant excessively backwards (Fig. 7(a)) causing it to fall after transition between relevant poses. In contrast, with the GPLVM, the learnt latent space almost perfectly reconstructed the demonstrated postures and, in high confidence regions, avoided unstable positions.

We applied RL in the learnt latent space with the following changes to the parameters: we replaced the Gaussian reward function (12) with a more pointed Laplacian[3] $R(\mathbf{x}) = \exp\{-20\|\mathbf{x}-\mathbf{x}^*\|\}$, set $\gamma = 0.995$, $\delta t = 0.5$, allowed for more stochastic action selection $\beta = 10$ and reduced the number of steps per episode to 200. An episode was aborted when an obstacle was hit, or when the predictive variance of a latent point was larger than 0.0004 (corresponding to standard deviation of 1.15 degrees in each joint). The latter criterion is an indirect measure of constraint fulfilment and replaces the direct measures from the previous experiments as they are unavailable in this completely unsupervised setting (where the only information about the task is given indirectly by the demonstrations themselves). In Fig. 7 we present the results.

As in the previous examples, RL consistently learnt good approximations of the value function and resulting policies moved the hands of the robot to the target while avoiding the obstacle. In the accompanying video we present these results on the real robot. We show an example demonstration, explore the resulting latent space online and execute trajectories of the learnt policy (see also Fig. 8).

V. Conclusion

In this paper, we explored the potential use of DR as abstraction for RL to improve its scalability to high-dimensional continuous spaces. Our hypothesis was that for constrained problems DR provides an alternative state representation, that exploits the hidden low-dimensional structure of the task. This benefits RL by (i) reducing the size of the space in which planning is done, and; (ii) avoiding wasteful exploration of the parts of the space in which the constraints are not satisfied and no solution exists. These benefits were evident in our experiments where RL in latent spaces emphatically outperformed learning in the original state space of the problem. By using this approach we saw that RL becomes feasible even in very high-dimensional, continuous systems such as the KHR-1HV humanoid to which the used RL method could otherwise not be applied.

For future work, we are looking into extending the approach in [7] to the nonlinear case within our framework to provide the DR with additional information about the relevance of demonstrated postures to the task given by the reward. Furthermore, we aim to reduce the number of episodes needed during RL training by employing more sophisticated RL techniques.

[3]The reward was defined over the positions of the hands and used the forward kinematics to evaluate the generated movements in simulation.

References

[1] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[2] F. Guenter, M. Hersch, S. Calinon, and A. Billard, "Reinforcement learning for imitating constrained reaching movements," *Adv. Robotics*, vol. 21, pp. 1521–1544, 2007.

[3] A. Barto and S. Mahadevan, "Recent advances in hierarchical reinforcement learning," *Discrete Event Dyn. Sys.*, vol. 13, pp. 341–379, 2003.

[4] G. Konidaris and A. Barto, "Skill discovery in continuous reinforcement learning domains using skill chaining," in *NIPS*, 2009.

[5] L. Li, T. Walsh, and M. Littman, "Towards a unified theory of state abstraction for MDPs," in *Proc. 9th Int. Symp. A.I. and Mathematics*, 2006.

[6] J. Morimoto and C. Atkeson, "Nonparametric representation of an approximated poincaré map for learning biped locomotion," *Auton. Robots*, vol. 27, pp. 131–144, 2009.

[7] J. Morimoto, S.-H. Hyon, C. G. Atkeson, and G. Cheng, "Low-dimensional feature extraction for humanoid locomotion using kernel dimension reduction," in *ICRA*, 2008.

[8] C. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[9] F. E. Udwadia and R. E. Kalaba, *Analytical Dynamics: A New Approach*. Cambridge University Press, 1996.

[10] K. Grochow, S. Martin, A. Hertzmann, and Z. Popovic, "Style-based inverse kinematics," in *SIGGRAPH*, 2004.

[11] R. Chalodhorn, K. MacDorman, and M. Asada, "Humanoid robot motion recognition and reproduction," *Adv. Robotics*, vol. 23, pp. 349–366, 2009.

[12] M. Howard, S. Klanke, M. Gienger, C. Goerick, and S. Vijayakumar, "A novel method for learning policies from variable constraint data," *Auton. Robots*, vol. 27, pp. 105–121, 2009.

[13] C. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[14] N. Lawrence, "Probabilistic non-linear principal component analysis with gaussian process latent variable models," *J. Machine Learning Research*, vol. 6, pp. 1783–1816, 2005.

[15] G. Neumann, "The reinforcement learning toolbox, reinforcement learning for optimal control tasks," Master's thesis, Graz Univ. Technology, 2005.