Task Space Retrieval Using Inverse Feedback Control

Nikolay Jetchev NIKOLAY.JETCHEV@FU-BERLIN.DE
Marc Toussaint MARC.TOUSSAINT@FU-BERLIN.DE
Machine Learning and Robotics Lab, FU Berlin, Arnimallee 7, 14195 Berlin, Germany

Abstract

Learning complex skills by repeating and generalizing expert behavior is a fundamental problem in robotics. A common approach is learning from demonstration: given examples of correct motions, learn a policy mapping state to action consistent with the training data. However, the usual approaches do not answer the question of what are appropriate representations to generate motions for specific tasks.Inspired by Inverse Optimal Control, we present a novel method to learn latent costs, imitate and generalize demonstrated behavior, and discover a task relevant motion representation: Task Space Retrieval Using Inverse Feedback Control (TRIC). We use the learned latent costs to create motion with a feedback controller.We tested our method on robot grasping of objects, a challenging high-dimensional task. TRIC learns the important control dimensions for the grasping task from a few example movements and is able to robustly approach and grasp objects in new situations.

1. Introduction

Imitation learning is an important tool for training robots in complex tasks (Argall et al., 2009). Using machine learning techniques to find structure and learn policies from demonstrations of desired behavior is often much more efficient than hand-crafting robot motion controllers. However, the utility of the behaviors learned in this way is still limited by the implicit assumptions made by the human designers. The question of "what to imitate", i.e. which aspects of the observed motions should be duplicated, is not answered in general.

Appearing in *Proceedings of the 28th International Conference on Machine Learning*, Bellevue, WA, USA, 2011.

This issue is particularly challenging in the context of articulated robotics: to generate complex movements for a certain task, a controller typically minimizes costs in a special movement representation or with respect to multiple task features simultaneously (e.g., collision avoidance, hand positioning, finger alignment, etc). Which features are suitable and how they are weighted depends on the task at hand. The target of a good motion is typically not a specific configuration state, but a whole task manifold. The challenge in imitation learning thus becomes to retrieve the latent movement representation (which task features are used in the controller) rather than to repeat a point-to-point movement. As an example consider a robot grasping an object: from observing successful motions we should learn that some finger configurations relative to the object surface result in good motions and use such representations in the controller, rather than fix targets in the direct robot configuration space.

Task Space Retrieval Using Inverse Feedback Control (TRIC) addresses the above issues by discovering latent costs and latent features of these costs in data and using them to generate motions, which has several advantages. First, it can handle example demonstrations in high dimensional spaces and select the important features for movement. Second, it can generalize well to situations and constraints unseen during demonstrations (e.g. grasping objects on different positions and collision avoidance with new obstacles), because the learned cost function defines a task manifold.

1.1. Related Work

Imitation learning via Direct policy learning (DPL) (Pomerleau, 1991) is a popular approach for learning from demonstration. In essence DPL takes demonstration data in the form of motion trajectories and uses supervised learning to estimate a controller that would reproduce the trajectories. This works well when the exact motion reproduction is desired (gestures, point-to-point motions), but it can have difficulties to generalize and modify the behaviors in new situations. Dy-

namic Movement Primitives (Schaal et al., 2003) are an advanced method that learns parameters of a dynamic system which defines the controller. However, it still requires to pre-specify a task space in which the attractor operates.

Inverse Optimal Control (IOC) (Ratliff et al., 2006), also known as Inverse Planning or Inverse Reinforcement Learning, takes demonstrations in some state space, and learns a state cost function that gives rise to a policy consistent with this data. IOC provides a general framework to retrieve latent objectives (rewards or costs) in observed behavior. (Ratliff et al., 2009) describes a combination of IOC and DPL.

Our approach TRIC jointly addresses the problems of finding relevant features of the motion, learning the behavior to imitate, and retrieving latent costs leading to such behavior. This is a different approach for motion feature selection than looking at data variance in an unsupervised way (Jenkins & Matarić, 2004).

(Muehlig et al., 2009; Billard et al., 2004) examine different task spaces for robot motions and select the best ones for reproducing different tasks. However, both examine only a small pool of possible task spaces, whereas our method can find rich motion representations from high dimensional task spaces.

Grasping is a challenging task, which has received attention in the context of learning by demonstration and robotics. (Tegin et al., 2009) acquires grasps from human movements and repeats them in joint space. (Kroemer et al., 2010) uses the hand fingertips for grasping within the Dynamic Motion Primitives framework. (Gienger et al., 2008) creates a task manifold of presampled grasp positions and orientations and a motion potential towards them. None of the above works try to extract from data a representation for grasping.

This paper will proceed with background on methods we compare with or build on, namely DPL, IOC and discriminative learning. In Section 3 we will present our algorithm and give details of the loss function used to train a cost function and how it is used for motion control. We test the effectiveness of our method in a robot grasping application in Section 4.

2. Background

2.1. Direct Policy Learning

A standard way to describe a robot trajectory is $\{x_t, u_t\}_{t=0}^{T}$, where x_t represents the robot state at time t and u_t is the control signal (e.g. the rate of change $\dot{x_t}$). DPL tries to find a policy $\pi : x_t \mapsto u_t$ from these observations. Different assumptions can be made for the choice of x,u and π (Pomerleau, 1991; Calinon & Billard, 2007), with refinements like data transformations and active learning. Given a parameterization of the policy, DPL essentially corresponds to a regression problem, e.g. with loss:

$$E_{dpl} = \sum_{t=0}^{T} \|\pi(x_t) - u_t\|^2 \tag{1}$$

where $\|.\|$ denotes the L_2 norm. Minimizing E_{dpl} finds a policy close in the least squares sense to the demonstrations. The above loss can be extended to multiple demonstration trajectories by averaging over them.

(Howard et al., 2009) introduces an interesting alternative loss for DPL:

$$E_{inc} = \sum_{t=0}^{T} (\|u_t\| - \pi(x_t)^T u_t / \|u_t\|)^2 \tag{2}$$

This loss penalizes the discrepancy between the projection of the policy $\pi(x_t)$ on u_t and the true control u_t. The motivation is that it can adapt robustly to constraints (e.g. collisions with objects) unseen in the demonstrations because of implicit regularization.

When the state and control spaces are high dimensional DPL has a disadvantage: generalization is an issue and would essentially require the data to cover all possible situations. Our approach aims to improve generalization by extracting the relevant task features from data.

2.2. Inverse Optimal Control

Inverse Optimal Control (Ratliff et al., 2006) assumes that policies π give rise to expected feature counts $\mu(\pi)$ of feature vectors ϕ, often within a Markov Decision Process framework. A weight vector w is learned by the following loss, such that the behavior demonstrated by the expert π^* has higher expected reward (negative costs) than any other policy:

$$\min \ \|w\|^2 \tag{3}$$

$$s.t. \ \forall \pi \ \ w^T \mu(\pi^*) > w^T \mu(\pi) + \mathcal{L}(\pi^*, \pi) \tag{4}$$

The term $w^T \mu(\pi)$ defines an expected reward, linear in the features. The scalable margin $\mathcal{L}$ penalizes those policies that deviate more from the optimal behavior of π^*. The above loss can be minimized with a max margin formulation. Efficient methods are required to find the π that violates the constraints the most and add it as new constraint.

Once the reward model is learned, another module is required to generate motions maximizing the reward,

e.g. (Ratliff et al., 2006) uses an A^* planner to find a path to a target with minimal costs.

Learning a policy based on estimated costs is much more flexible than DPL, and a simple cost function can lead to complex optimal policies. In some domains it is much easier to learn a mapping from state to cost, than a mapping from state to action which is a more complex and higher dimensional problem, especially when considering actions in high dimensional continuous spaces such as robot control. IOC can also often generalize better to new situations, because states with low costs create a task manifold.

2.3. Discriminative Learning

We will propose a method related to IOC, using a discriminative learning framework. Energy based models (LeCun et al., 2006) provide a common framework for many learning problems, including structured output regression. Data is given of the form $\{x_i, y_i\}$: pairs of input and output values. As in standard discriminative approaches (e.g., structured output learning), the energy or cost $f(x_i, y_i; w)$ provides a discriminative function such that the true output should get the lowest value from the model f:

$$y_i = \arg\min_{y \in \mathcal{Y}} f(x_i, y) \tag{5}$$

Training the parameter vector w of the model f is done by minimizing a loss over the dataset. The loss should have the property that f is penalized whenever the true answer y_i has higher energy than the false answer with lowest energy which is at least distance r away:

$$\tilde{y}_i = \arg\min_{y \in \mathcal{Y}, \|y - y_i\| > r} f(x_i, y) \tag{6}$$

Instead of using the common hinge loss $L_{hinge}(x_i, y_i) = \max(0, m + f(x_i, y_i) - f(x_i, \tilde{y}_i))$ we choose the alternative log loss:

$$L_{log}(x_i, y_i) = \log(1 + e^{f(x_i, y_i) - f(x_i, \tilde{y}_i)}) \tag{7}$$

which is a soft form of L_{hinge} with infinite margin m. Finding the most offending answer $\tilde{y}_i$ is very often a complicated inference problem in itself.

3. Task Space Retrieval Using Inverse Feedback Control

The essence of our algorithm is to discover latent costs f that characterize the demonstrations: the teacher movements always decrease f, which can be viewed also as a motion potential. A central idea in our approach is to model f as a "sparse" function of a high-dimensional feature vector y, which offers a large variety of potential geometric features that might be relevant for a motion. The optimization of f implies a choice of these features. In Section 3.1 we describe our motion model, that is, how f implies the movement. In Section 3.2 we define a training loss for learning f from demonstrations. The terms of the loss are explained in Sections 3.3 and 3.4.

3.1. Robot Motion Model

We denote the vector of joint angles of the robot body by $q \in \Re^n$. A robot movement trajectory for T time steps is $\{q_i\}_{i=1}^T$. We assume that for each joint configuration q we can compute a high-dimensional feature vector $\phi(q) \in \Re^d$. The feature vector will typically comprise all possible relative and absolute positions and distances between all landmarks on the robot and external objects—clearly, the size of the feature vector is combinatorial in the number of objects and body parts. It is the objective of learning to select relevant features to describe the latent cost function f of a motion, see Figure 1. The features $\phi(q)$ are non-linear in q and can be computed from the robot kinematics. E.g. if the task is to move the robot hand to a target position, a properly chosen feature that codes this distance will allow much easier control than the robot joint configuration space. TRIC uses the richness of the representation $\phi(q)$ to learn a cost model and select important features.

As in IOC, the learned cost function $f : \phi(q) \mapsto \Re$, or equivalently $f \circ \phi : q \mapsto \Re$, determines the motion (policy) of the system. We assume that, given f, the robot generates motion with a method commonly used in robotics: motion rate control. More precisely, the robot motion is generated by imposing a smooth decreasing "motion" on f which is translated back to joint angle motions using inverse kinematics (IK).

More formally, motion rate control can be defined as computing a new robot pose $q_{t+1} = \arg\min_q C(q)$ in each iteration by minimizing the objective function

$$C(q) = \|q - q_t\|^2 + \delta_1 \|f \circ \phi(q) - f \circ \phi(q_t) + \delta_2\|^2 + C_{\text{prior}}(q) \tag{8}$$

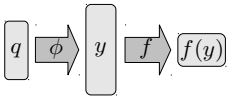


Figure 1. Motion representation scheme: the joints q are mapped by ϕ to a high-dimensional feature vector y, which is then used to find costs $f(y)$. In general y has many more dimensions than q and f should be sparse in y.

where the first term penalizes step length, the second term aims for a decrease in $f \circ \phi$ by a rate δ_2, and the third term imposes additional standard costs for joint limits and collisions. The solution – assuming local linearization of $f \circ \phi$ and neglecting C_{prior} for simplicity – is the standard IK equation

$$q_{t+1} = q_t - \lambda_t \mathcal{J}^\sharp(q_t) = q_t - \frac{\lambda_t}{\|\mathcal{J}\|^2}\mathcal{J}(q_t) \tag{9}$$

where $\mathcal{J}^\sharp$ is the pseudoinverse of $\mathcal{J}$ and λ_t is some positive scalar. Since f is scalar and $\mathcal{J}$ is a vector, it holds that $\mathcal{J}^\sharp = \mathcal{J}/\|\mathcal{J}\|^2$. The derivative of $f \circ \phi$ with respect to q is

$$\mathcal{J} = \frac{\partial f \circ \phi}{\partial q} = \frac{\partial f}{\partial \phi}\frac{\partial \phi}{\partial q} \tag{10}$$

Iteratively making steps $q_t, q_{t+1}, ...$ with this motion model generates a continuous motion trajectory decreasing the costs $f \circ \phi(q) = f(y)$. Note that our goal is not to find global optima of the costs.

3.2. Training Loss for Cost f

The problem now becomes to learn costs f such that our motion model generates motions that correspond to the demonstrated behaviors. Let $D = \{q_t^i, y_t^i, \frac{\partial \phi}{\partial q}(q_t^i)\}_{i,t}^{N,T}$ be a set of N demonstration trajectories of fixed length T time steps. The joint space movements are projected to the feature space using $\phi(q_t^i) = y_t^i$.

We assume that f is parameterized by parameters w – in Section 4 we will specify a specific parameterization mixing linear and non-linear components, which we evaluate in the experiments. We train the cost function f based on the following loss:

$$L(D; w) = \sum_{i=1}^{N}\sum_{t=1}^{T}(\alpha_n L_n(y_t^i; w) + \alpha_g L_g(q_t^i; w)) + \alpha_w \|w\|_1 \tag{11}$$

The hyperparameters $\alpha = \{\alpha_n, \alpha_g, \alpha_w\}$ determine the influence of the different loss terms. The L_1 regularization term $\|w\|_1$ in Equation (11) forces sparsity in the parameters and indirectly performs feature selection with respect to y. Because of the coupling $f \circ \phi$, the sparsity of f means that the motion rate control will lead to joint states changing some task dimensions of y and not caring about others – which we call task space retrieval and selection. Given the overall loss $L(D; w)$ we use gradient-based optimization to optimize with respect to the parameters w.

The next sections will explain in more detail L_n and L_g, and how they influence the motion cost f.

3.3. Discriminating Teacher Demonstrations via Loss Term L_n

For optimal control we need a cost function f that is consistent with the observed trajectories: the demonstrations are assumed to be near optimal and have low costs, which discriminate them from all other possible movements. The usual approaches to discriminative learning would require finding the most offending false answer during the loss minimization as in Equation (6). However, in our case this is expensive since it requires calls to the robot simulator to find $\arg\min f \circ \phi(q)$ repeatedly. To speed up training we create a small set of $m = 1, .., M$ of synthetic "noisy" samples which need to be discriminated from the demonstrated trajectories:

$$\tilde{q}_{t,m} = q_{t-1} + \mathcal{N}(0, \sigma) \tag{12}$$

$$\tilde{y}_{t,m}^i = \phi(\tilde{q}_{t,m}) \tag{13}$$

The noisy joint states $\tilde{q}_t$ are created by adding Gaussian noise to the robot joint configuration of the previous time slice q_{t-1}. We add noise on q and not on y directly since the states y where the robot can be are a manifold constrained by the robot kinematics. We define weights equal to the distance to the true demonstration trajectory *ahead* in time:

$$\epsilon_{m,t}^i = \min_{\gamma \in \{0, T-t\}} |q_{t+\gamma}^i - \tilde{q}_{t,m}^i|_2 \tag{14}$$

Samples that are near a correct state $q_{t+\gamma}$ in the future will have low weights. Samples that are away from the true trajectory, or near some state $q_{t-\gamma_0}$ back in time will have high weights and their loss contribution will be higher. Such sample weighting arises from the assumption that the demonstrations in our motion model move to states of lower cost as time progresses, implicit in our motion model. The way we create these synthetic samples and weight them is illustrated in Figure 2.

The term L_n is then defined as:

$$L_n(y_t^i; w) = \sum_{m=1}^{M} \epsilon_{m,t} \log(1 + e^{f(y_t^i; w) - f(\tilde{y}_{t,m}^i); w)}) \tag{15}$$

An intuitive interpretation is that we "push down" the energy $f(y)$ of the true samples and "push up" the energy of the artificial noisy samples, proportional to their weights as in Figure 3, where for simplicity $q = y \in \Re^2$. We use log loss similar to Equation (7).

Equation (15) has similarities with IOC and Equation (3): the expected state features are our y, the comparison with all policies π is replaced with comparing the true sample y_t and the noisy data points $\tilde{y}_t$, and the

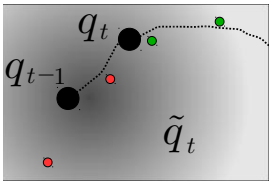


Figure 2. The noise $\tilde{q}_t$ comes from a Gaussian centered at q_{t-1}. A few "noisy" samples are shown, colored by their weights $\epsilon_{m,t}$: low weights are green and high weights red.

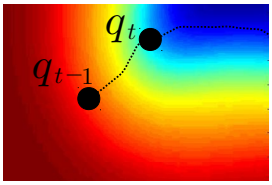


Figure 3. The loss term L_n pushes down the costs $f \circ \phi(q)$ near the true trajectory samples q_t , and up the cost of the generated noise. Red represents high cost.

scalable margin is the weight ϵ, which makes the cost margins dependent on the distance of the false vector to the experts' demonstration.

3.4. Making the Derivative $\mathcal{J}$ Consistent with the Demonstrations via term L_g

Our motion model (9) implies that the gradient $\mathcal{J}$ should align with the relative steps of the demonstrated trajectories in joint space. The second loss term L_g reflects this property and is defined as:

$$L_g(q_t^i; w) = \frac{\mathcal{J}(q_t^i)^T}{\|\mathcal{J}(q_t^i)\|} \frac{(q_t^i - q_{t-1}^i)}{\|q_t^i - q_{t-1}^i\|} \tag{16}$$

where $\mathcal{J}(q_t^i)^T$ the gradient of $f \circ \phi$.

Note that computing $\dfrac{\partial L_g}{\partial w}$ involves calculating

$$\frac{\partial \mathcal{J}}{\partial w} = \frac{\partial^2 f}{\partial \phi \partial w} \frac{\partial \phi}{\partial q} \tag{17}$$

The first term on the right side can be calculated from f, and the second is the so-called kinematic Jacobian.

It can be seen that L_g is related to (Howard et al., 2009) and the term of Equation (2): it is minimized by the same policy if we consider the normalized gradient $\frac{\mathcal{J}(q_t^i)^T}{\|\mathcal{J}(q_t^i)\|}$ as control policy $\pi(q_t)$ and the control signals $u_t = \frac{q_{t+1} - q_t}{\|q_{t+1} - q_t\|}$ are normalized to constant length.

4. Experiments

The task we examine is grasping of objects by a robot. Our robot is the Schunk LWA 3 arm with 7 Degrees of Freedom (DoF) and Schunk SDH hand with 7 DoF, making a joint configuration space $q \in \Re^{14}$. We use as demonstration source the method of (Dragiev et al., 2011), which is an efficient human-designed controller to grasp objects.The setup consists of the robot and a grasp target.

We generate a dataset for training by translating the position of the target object a_1 in a regular grid $50 \times 40 \times 40$cm in front of the robot, taking 3 positions in each grid dimension, leading to $N = 27$ different settings and training trajectories. Each grasp movement is generated for 5 seconds, and we sampled $T = 40$ time steps from it, once every 125ms. The target is grasped faster when the object is closer to the robot, but TRIC easily handles trajectories not aligned in time or with different durations. Unless stated otherwise we generate $M = 60$ random samples by randomly sampling collision-free joint configurations with a normal distribution with $\sigma = 0.05$, corresponding to 0.05 radians joint angle standard deviation.

4.1. Choice of High Dimensional Features

We construct a rich feature space representation through a set of 8 landmarks on the robot and the grasp target object $A = \{a_i\}_{i=1}^8$, shown in Figure 4. This includes the center of the target object a_1, the three fingertips a_2, a_3, a_4, the three lower digits a_5, a_6, a_7, and the palm center a_8. We denote the positions of landmark a relative to the frame of object b as x_a^b. The feature space $y = \phi(q)$ consists of pairwise landmark positions and their norms:

$$y = \{x_a^b, \|x_a^b\|\}_{a \in A, b \in A, a \neq b} \in \Re^{224} \tag{18}$$

These are chosen as a reasonable set to capture grasp poses, but much larger feature spaces are possible.

We preprocess each dimension of y by rescaling it to $[0, 1]$. Some features are redundant due to the specific

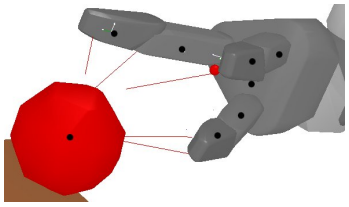


Figure 4. The landmark set A used to generate the features y indicated by black dots.

geometry of the landmarks. We remove the redundant features using correlation as a measure (Haindl et al., 2006), resulting in the final motion features $y \in \Re^{100}$.

The parametric form of the costs f is a sigmoid neural network with K nodes combined with a linear term:

$$z = \frac{1}{1 + e^{W_1 y}} \tag{19}$$

$$f(y; w) = z^T W_2 + y^T W_3 \tag{20}$$

The parameters w consist of $W_1 \in \Re^{K \times d}$, $W_2 \in \Re^K$, $W_3 \in \Re^K$ and $z \in \Re^K$ is the hidden layer.

With this model we let the loss minimization training decide on a trade-off between the linear and non-linear cost function terms. We trained our models for 100 iterations with the BFGS method from the MATLAB Optimization Toolbox. The training time scales with K: for a linear model $K = 0$ it takes less than a minute, and 30 minutes for $K = 30$.

We used Equation (9) for motion rate control using the learned costs $f \circ \phi(q)$. We rescaled W_2 and W_3 by dividing by $|W_2| + |W_3|$, which does not change the motion implied by the cost function, and allows for better comparison of differently trained models. We set δ_1 to 1000 and δ_2 to 0.03, as a reasonable motion rate. We used the standard additional constraints on collision avoidance and joint limit avoidance coded in the term C_{prior}, with weights to ensure that the IK controller stays collision free and within joint limits.

4.2. Results

We use the errors of the task variables defined in (Dragiev et al., 2011) as a validation metric for grasp quality: it requires that all fingers lie at the object surface and oppose each other. This metric evaluates the final grasping posture of motion with a number between 0 and 1, where values below 0.25 are good grasps. We show that our method TRIC learns an underlying cost function of the demonstrated task. We create a new set of 15 trajectories on random positions within the training grid and evaluated the grasping costs of different TRIC models with 5 seconds for execution. The default values of the hyper-parameters from Equation (11) are $\alpha_n = 2, \alpha_g = 0.4, \alpha_w = 0.001$, and the other parameters are $M = 60$ samples, $N = 27$ trajectories, $K = 30$ sigmoids for f.

In a first experiment we test 4 variations of the TRIC model by switching on and off the term L_g from Equation (10) and comparing a pure linear model ($K = 0$) to a nonlinear network ($K = 30$). The setting of linear f and no derivative consistency term L_g is similar to a classical IOC method.

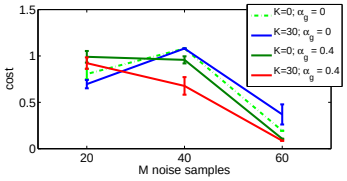


Figure 5. The effect of changing M on TRIC.

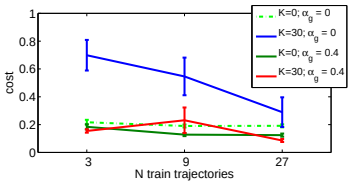


Figure 6. The effect of changing N on TRIC.

Figure 5 shows how sensitive the 4 methods are to the number of samples M. More samples allow better training of f, but with 60 samples the performance is close to the teacher. We also investigated varying the number of training trajectories N, by creating smaller sets of 3 and 9 trajectories. Figure 6 shows that with as few as 3 trajectories TRIC could learn grasping. Overall, the parameter M seems more critical than N, however the mediocre performance of TRIC with $K = 30$ at $N = 9$ indicates that non-linear neural networks are sensitive to random initialization and small training set.

From both Figures 5 and 6 we may conclude that more complex models for f with $K = 30$ sigmoids can learn better than pure linear models, however with the drawback of longer training time. The term L_g implying the derivative constraint is beneficial and $\alpha_g = 0.4$ leads to better performance. The performance of the *Teacher* from (Dragiev et al., 2011) was 0.06 ± 0.01 on the validation set, and the best TRIC models had similar performance with 0.08 ± 0.01.

In a second experiment we implemented different baseline DPL methods for comparison, using ϕ for state space x, and $q_{t+1} - q_t$ for control space u (using the high-dimensional features y for control was ineffective), and a neural network with 60 sigmoids to learn π. Our results was that DPL with loss Equation (2) could not learn the train data. DPL with loss Equation (1) had poor performance on the validation data. The controller following $\pi(x_t) = u_{t+1}$ had problems with generalization and instability, and resulted on sub-optimal grasps even on the training set.

To test this more quantitatively we defined a new, simpler validation set, consisting of (a) 30 scenarios from

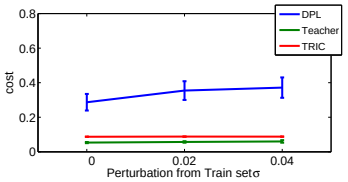
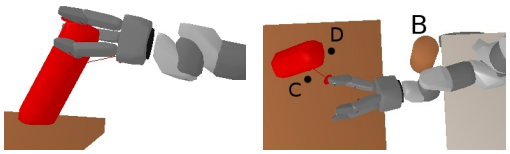


Figure 7. Comparison to Direct Policy Learning with increasing "distance" between training and test data.



(a) Grasping pose for cylinder. (b) Grasp the target avoiding obstacle B.

Figure 8. Implicit obstacle avoidance and cylinder grasping with TRIC.

the original train set, (b) 30 scenarios with added random translation of the target position distributed as a Gaussian with 2cm standard deviation, or (c) 30 scenarios created analogously with 4cm standard deviation. Figure 7 shows how the results degrade for DPL as the validation scenarios become remote from the training samples, whereas TRIC remains robust.

In a third experiment we test TRIC on more complex objects. We create a new set of 10 demonstrations, by rotating an elongated cylinder (30cm height and 5cm radius) on its Y-axis by values uniform in $[0, \pi/2]$ radians. We then train TRIC with the default parameters and test the learned f on grasping of cylinders rotated differently than the train set, see Figure 8(a). The performance of TRIC with $K = 30$ (0.09 ± 0.02) is similar to *Teacher* (0.07 ± 0.01). Linear TRIC has cost 0.19 ± 0.02, thus, more complex motion policies require non-linear models for f.

In a fourth experiment we examine how TRIC reacts to an obstacle B in its path to the target—since our motion model include a prior cost term avoiding obstacles TRIC can in principle handle such cases even when not present in the training set: in our evaluation B appears *only* in the validation set, not in the train set. TRIC is reasonably robust: the motion rate controller stays out of collisions and a straight approach to pose D is no longer feasible, but the grasping pose C out of the many poses with low cost can be found, see Figure 8(b). Clearly, this implicit local collision avoidance behavior has limitations. In more complex cluttered scenes the costs f might be used in a global planner to avoid local minima.

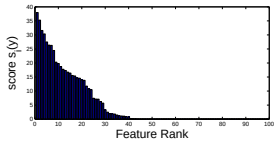


Figure 9. Effect of L_1-regularization: the features sorted by score $s(y)$.

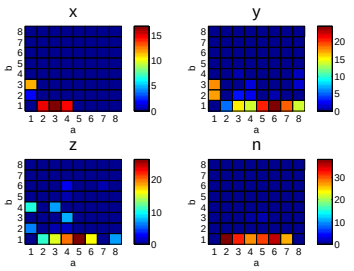
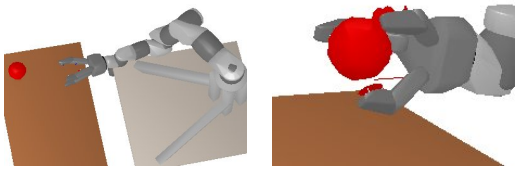


Figure 10. A map of the feature scores $s(y)$ represented as pairwise distances. One plot each for the three spatial dimensions (x,y,z) and the norm $n = \|x_a^b\|$. Columns code index a and rows index b of the two compared landmarks.

4.3. Motion Feature Selection

To get an insight into the feature selection done implicitly by the loss minimization of Equation (11), we defined a score for each feature equal to a weighted sum of the absolute values of coefficients of f from Equation (20): $s(y) = |W_1^T||W_2| + |W_3|$. In Figure 9 we display these scores on the 100 non-redundant dimensions of y. It can be seen that the L_1-regularization selected about 40 of these features for the cost f.

Figure 10 shows the feature scores as they relate to pairwise distances x_a^b in Equation (18). It can be seen that the features measuring relative distance of the target object from the fingers are the most important for the grasping task, which is a meaningful way to construct features for a grasping task.



(a) First approach the red target object and open the fingers (b) Finally align the fingers close to the object's surface

Figure 11. Illustration of the overall grasping motion.

5. Conclusion

In this paper we presented a novel method for learning a latent cost function from demonstrations, finding relevant task dimensions and generating motion. TRIC can generalize to different situations unseen during training. We tested its performance on a robot grasping task, presented results showing the effect of different training and model settings, and visualized the extracted motion features.

Our method has some limitations. First, it is not suitable for periodic motions and would need to add state information for previous time slices to deal with more complex motions. Second, we assume an accurate robot kinematic model and TRIC is sensitive to noise in the input features.

Promising future work is to couple TRIC with more interesting object representations like implicit surfaces. Further, the analysis of human demonstrations with TRIC may uncover interesting results concerning which features underly human motion generation.

6. Acknowledgments

This work was supported by the German Research Foundation (DFG), Emmy Noether fellowship TO 409/1-3. We would like to also thank the anonymous reviewers for their feedback.

References

Argall, Brenna D., Chernova, Sonia, Veloso, Manuela, and Browning, Brett. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57:469–483, May 2009.

Billard, A., Epars, Y., Calinon, S., Cheng, G., and Schaal, S. Discovering optimal imitation strategies. *Robotics and autonomous systems, Special Issue: Robot Learning from Demonstration*, 47(2-3): 69–77, 2004.

Calinon, Sylvain and Billard, Aude. Incremental learning of gestures by imitation in a humanoid robot. In *HRI '07: Proceedings of the ACM/IEEE international conference on Human-robot interaction*, pp. 255–262, 2007.

Dragiev, Stanimir, Toussaint, Marc, and Gienger, Michael. Gaussian process implict surface for object estimation and grasping. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2011.

Gienger, Michael, Toussaint, Marc, Jetchev, Nikolay, Bendig, Achim, and Goerick, Christian. Optimization of fluent approach and grasp motions. In *8th IEEE-RAS International Conference on Humanoid Robots*, 2008.

Haindl, Michal, Somol, Petr, Ververidis, Dimitrios, and Kotropoulos, Constantine. Feature selection based on mutual correlation. In *CIARP*, pp. 569–577, 2006.

Howard, Matthew, Klanke, Stefan, Gienger, Michael, Goerick, Christian, and Vijayakumar, Sethu. A novel method for learning policies from variable constraint data. *Autonomous Robots*, 27:105–121, 2009.

Jenkins, Odest Chadwicke and Matarić, Maja J. A spatio-temporal extension to isomap nonlinear dimension reduction. In *21st Int. Conf. on Machine Learning (ICML)*, 2004.

Kroemer, Oliver, Detry, Renaud, Piater, Justus H., and Peters, Jan. Grasping with vision descriptors and motor primitives. In *ICINCO (2)*, pp. 47–54, 2010.

LeCun, Yann, Chopra, Sumit, Hadsell, Raia, Ranzato, Marc'Aurelio, and Huang, Fu-Jie. A tutorial on energy-based learning. In *Predicting Structured Data*, 2006.

Muehlig, Manuel, Gienger, Michael, Steil, Jochen J, and Goerick, Christian. Automatic selection of task spaces for imitation learning. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pp. 4996–5002, 2009.

Pomerleau, Dean A. Efficient training of artificial neural networks for autonomous navigation. *Neural Comput.*, 3:88–97, 1991.

Ratliff, Nathan, Ziebart, Brian, Peterson, Kevin, Bagnell, J. Andrew, Hebert, Martial, Dey, Anind K., and Srinivasa, Siddhartha. Inverse optimal heuristic control for imitation learning. In *Proc. of AISTATS*, pp. 424–431, 2009.

Ratliff, Nathan D., Bagnell, J. Andrew, and Zinkevich, Martin A. Maximum margin planning. In *26th Int. Conf. on Machine Learning (ICML)*, pp. 729–736, 2006.

Schaal, Stefan, Peters, Jan, Nakanishi, Jun, and Ijspeert, Auke Jan. Learning movement primitives. In *International Symposium on Robotics Research*, pp. 561–572, 2003.

Tegin, Johan, Ekvall, Staffan, Kragic, Danica, Wikander, Jan, and Iliev, Boyko. Demonstration-based learning and control for automatic grasping. *Intelligent Service Robotics*, 2:23–30, 2009.