Planning and Moving in Dynamic Environments
A Statistical Machine Learning Approach

Sethu Vijayakumar[1], Marc Toussaint[2], Giorgios Petkos[1], and Matthew Howard[1]

[1] School of Informatics, University of Edinburgh, Edinburgh EH8 9AB, UK
sethu.vijayakumar@ed.ac.uk
[2] Technical University of Berlin, 10587 Berlin, Germany
mtoussai@cs.tu-berlin.de

Abstract. In this chapter, we develop a new view on problems of movement control and planning from a Machine Learning perspective. In this view, decision making, control, and planning are all considered as an inference or (alternately) an information processing problem, i.e., a problem of computing a posterior distribution over unknown variables conditioned on the available information (targets, goals, constraints). Further, problems of adaptation and learning are formulated as statistical learning problems to model the dependencies between variables. This approach naturally extends to cases when information is missing, e.g., when the context or load needs to be inferred from interaction; or to the case of apprentice learning where, crucially, latent properties of the observed behavior are learnt rather than the motion copied directly.

With this account, we hope to address the long-standing problem of designing adaptive control and planning systems that can flexibly be coupled to multiple sources of information (be they of purely sensory nature or higher-level modulations such as task and constraint information) and equally formulated on any level of abstraction (motor control variables or symbolic representations). Recent advances in Machine Learning provide a coherent framework for these problems.

1 Introduction

It is sometimes asked why one should apply statistical or probabilistic methods in robotics when the environment is to a large degree deterministic. Are not classical deterministic methods perfectly suited? A first reply might point out that any realistic mechanical system also includes noise and that statistical approaches will increase robustness. While this is certainly true, we would emphasize another motivation for statistical or probabilistic approaches: the computational or information processing perspective.

The control and preparation of movement, the planning of behavior, and the interaction with natural environment – all, to a large extent, incorporate and exploit many pieces of information about the current state. This information either stems from direct observation (sensors), from internal estimations (like

B. Sendhoff et al. (Eds.): Creating Brain-Like Intelligence, LNAI 5436, pp. 151–191, 2009.
© Springer-Verlag Berlin Heidelberg 2009

the position of a tracked but occluded object, or the self-location in an internal map), or from internally induced constraints (such as goals or costs). In most general terms, decision making or planning involves combining all this information optimally in order to gain some perspective about the consequences of possible decisions.

In fact, a most significant characteristic of the central nervous systems (CNS) is that many pieces of information are represented in parallel on many different levels of representations. It seems that much "evolutionary effort" was put in developing such representations including properties such as their topographic organization. Without going into details about how all these pieces of information are processed to get a unified percept, it generally seems clear that they are coupled in the sense of sustaining some consistency and enabling transformations between each other. The discussion whether neural processes can faithfully be abstracted to implement basic information processing mechanisms such as inference is beyond the scope and aim of this paper.

Machine Learning (a field that in large parts has developed from the early "Neural Information Processing" approaches) has developed methods that exactly abstract this general kind of information processing. Thus, another answer to the introductory question is that the probabilistic and statistical framework allow us to grasp how to represent information, to fuse and process information, to cope with missing information, or to learn statistical dependencies that provide the basis for processing information and performing inference.

Machine Learning approaches explicitly distinguish between separate representations of information (e.g., random variables) and processes of inference. A most crucial question here is what are the representations of information that we should be concerned with that are relevant for a given task; i.e., which (latent) random variables should we introduce. This can be seen in close analogy to the question which representations are inherent in the brain, i.e., what information is represented by different areas. Within this view, we may distinguish four levels of problems:

- What are the variables (or representations) that are useful for the current situation, tasks or environment?
- How do these depend upon each other?
- What are the current goals, targets or constraints?
- What are their implications on the free variables – such as the immediate actions or motor commands?

Each of these levels requires some methods for:

1. Development/extraction of suitable representations (for e.g., expert knowledge or some algorithmic procedure).
2. Regression/modelling techniques to learn local contingencies (models) between coupled variables.
3. Conditioning and biasing of the random variables
4. Inference.

The outline of this chapter follows these different levels. In Sec. 2, we first reformulate classical control schemes in terms of Bayesian inference – as an example for steps 3 and 4. This approach can nicely be extended to movement planning problems going beyond classical solutions.

In Sec. 3, we first review efficient approaches for statistical learning (step 2) in the context of control. Extending this to the development of 'internal' representations under multiple contexts (in Machine Learning terms, the learning of latent variable models) is tackled next. We will demonstrate a basic exemplar for the extraction of a latent context variable from data; indeed, this can be thought of as tackling the scenario of missing information, for e.g., the explicit mass information of a load to be manipulated.

Finally, in Sec. 4, we consider the application of statistical learning in the context of imitation or apprentice learning. From demonstrated movement data, we can learn parameters of the movement prior (the so-called null-space potential). By learning this objective function that is implicit in the observed behavior, we can potentially reproduce the behavior on a functional level as opposed to merely copying a motor trajectory and exploit this in order to generalize to novel tasks and previously unseen constraints.

2 Planning and Control

In this section, we will contrast the optimization perspective of movement control and planning to the inference or information processing perspective – technically by relating an objective function to a corresponding probabilistic model. The reader might object that there is no difference anyway because any cost function can surely be translated to a likelihood function somehow such that the optimal solution in the first case is the maximum a posteriori (MAP) solution in the second. However, the optimization and the inference views differ conceptually and technically, as we shall see next.

First, the information processing view always computes distributions (posteriors) over the desired variables whereas the optimization view computes only a single solution at every stage. If only the MAP solution is of interest, this makes no difference. However, when some more information becomes accessible after the optimization or computation, then the information processing view can perfectly integrate this new information in the previous calculations whereas the optimization view would have to recompute the solution. In natural environments, the incremental and stochastic nature of information over time can be exploited better in the latter framework. In more general terms, the inference procedure generates a distribution which is open to be coupled to further information or modulated otherwise from a "higher level"; whereas the optimization view generates a single solution for which it is unclear how to couple it to new information, adapt or modulate it.

Second, the technical tools used for optimization versus inference differ significantly, for e.g., this might be quadratic programming versus belief propagation. In practice, either paradigms have associated pros and cons with respect

to computational cost, robustness and ease of implementation – as we will explore next.

2.1 Redundant Control as Optimization Problem

It turns out that a very general class of kinematic and dynamic control schemes (e.g., those described in [1]) can be understood as special case solutions to a simple optimization problem. Consider the cost function

$$L = ||x - Jq||^2_{C^{-1}} + ||q||^2_W + 2h^T q \tag{1}$$

where $||q||^2_W = q^T W q$ describes the Mahalanobis norm. The minimum of this function can easily be found by taking the derivative,

$$\frac{\partial L}{\partial q} = -2(x - Jq)^T C^{-1} J + 2q^T W + 2h^T = 0 \qquad \Rightarrow \tag{2}$$

$$\begin{aligned} q &= (J^T C^{-1} J + W)^{-1}(J^T C^{-1} x - h) \qquad (3)\\ &= W^{-1} J^T (JW^{-1} J^T + C)^{-1} x - [\mathbf{I} - W^{-1} J^T (JW^{-1} J^T + C)^{-1} J]\, W^{-1} h \end{aligned}$$

The last equation uses the Woodbury identity:

$$J^T C^{-1} J + W)^{-1} J^T C^{-1} = W^{-1} J^T (JW^{-1} J^T + C)^{-1}, \tag{4}$$

which generally holds for any positive definite C and W.

In Machine Learning this minimization is well-known as ridge regression. In the more typical notation rewritten as "$||y - X\beta||^2_{C^{-1}} + \lambda||\beta||^2$", we are given a input data matrix X, an output data vector y with an aim to find the regressor β. The first term of L corresponds to the squared error (under noise covariance C) while the second term is a regulariser (or stabiliser) as introduced by Tikhonov and takes the form of a ridge $\lambda||\beta||^2$. For uniform output covariance $C = I$ and $h = 0$, (4) tells us that the solution to ridge regression is $\beta^* = X^T(XX^T + \lambda I)^{-1} y$.

In the case of redundant kinematic control (resolved motion rate control), one would like to compute joint velocities $\dot{q}$ which fulfill a lower-dimensional task constraint $\dot{x} = J\dot{q}$ while minimizing the absolute joint velocity $||\dot{q}||^2_W$ and following the negative gradient $h = -\nabla H(q)$. The solution is easily found from (4) for the limit $C \to 0$ (i.e., imposing zero variance in the task constraint):

$$\dot{q} = J^\sharp_W \dot{x} + (\mathbf{I} - J^\sharp_W J) W^{-1} \nabla H(q) \tag{5}$$

where $J^\sharp_W = W^{-1} J^T (JW^{-1} J^T)^{-1}$ is called the pseudo-inverse of J under the metric W.

In hierarchical motion rate control, one has a number of task variables $x_1, .., x_m$ with fixed priority. That is, the first constraint $\dot{x}_1 = J_1 \dot{q}$ needs to be fulfilled exactly, the second constraint $\dot{x}_2 = J_2 \dot{q}$ only within the nullspace of the first constraint, etc. The classical solution is given in [2,3]. It turns out that this scheme

can also be derived from the general solution of (4) when taking the cascaded limit $C \to 0$, where C is in block form and each block approaches zero with a different rate.

Finally, in dynamic control one would like to compute a control signal u which generates an acceleration $\ddot{q} = M^{-1}(u + F)$ such that a general task constraint $b = A\ddot{q}$ remains fulfilled while also minimizing the absolute norm $||u||^2_W$ of the control. Reformulating the constraint as $b - AM^{-1}(u + F) = 0$ and taking the limit $C \to 0$ (zero variance in the constraint), we can consult (4) and get the solution:

$$u = J^\sharp_W(b - JF) - (\mathbf{I} - J^\sharp_W J)W^{-1}h \ , \quad \text{with } J = AM^{-1} \tag{6}$$

which, for $h = 0$, is identical to Theorem 1 in [1]. Peters et al. discuss in detail important versions of this control scheme.

2.2 Redundant Control as an Inference Problem

There exists a straight-forward reinterpretation of the above control schemes in a Bayesian setting. At first sight, it might seem that nothing is gained from this reformulation. However, as we mentioned earlier, the information processing view can have significant practical and theoretical difference to the optimization view. For instance, the next section will derive an extension of the Bayesian control scheme to trajectory planning problem which departs from classical trajectory optimization algorithms.

Exploiting the fact that all terms in the loss function $L = ||x - Jq||^2_{C^{-1}} + ||q||^2_W$ are quadratic, we can formulate an equivalent likelihood based on Gaussian random variables as follows.

Let q_t be the n-dimensional robot configuration at time t and x_t some d-dimensional task variable. When discretizing time, we are interested in computing an actuator motion $\delta q = q_{t+1} - q_t$ of the robot given a desired task step $\delta x = x_{t+1} - x_t$. Instead of considering the deterministic 1st order approximation $\delta x \doteq J\delta q$, we assume that, at the particular current state, we have

$$P(\delta x \,|\, \delta q) = \mathcal{N}(\delta x, J\delta q, C) \tag{7}$$

where $\mathcal{N}(x, a, C)$ is the Gaussian density function over x with mean a and covariance matrix C. In the limit $C \to 0$, this corresponds to the deterministic 1st order approximation; non-zero C allow us to have a tolerance w.r.t. fulfilling the task constraint. We compute the posterior over δq *conditioned* on the desired step δx_i,

$$P(\delta q \,|\, \delta x) \propto P(\delta x \,|\, \delta q)\ P(\delta q) \ . \tag{8}$$

For this we need to specify a prior that we choose as $P(\delta q) = \mathcal{N}(\delta q, 0, W^{-1})$. Note that here we identify the precision matrix W directly with the regularizer metric W in (1). Using

$$P(\delta x \,|\, \delta q) = \mathcal{N}(\delta x, J\delta q, C) = \mathcal{N}(J\delta q, \delta x, C) \propto \mathcal{N}(\delta q, J^{-1}\delta x, J^{-1}CJ^{-1T}) \quad (9)$$

we derive the posterior over δq as

$$\begin{aligned} P(\delta q \,|\, \delta x) \qquad & = \mathcal{N}(\delta q, A^{-1}a, A^{-1}) \;, \\ \text{with} \quad a = J^T C^{-1}\delta x \;, \quad & A = W + J^T C^{-1} J \end{aligned} \quad (10)$$

Clearly, the MAP motion

$$\widehat{\delta q} = A^{-1}a = (W + J^T C^{-1} J)^{-1} J^T C^{-1} \delta x \quad (11)$$

coincides with standard redundant control for $C \to 0$ using the Woodbury identity. From this rather trivial result we may conclude the following

- The classical regularizer matrix W, which appears in the pseudo-inverse $J^\sharp_W$, plays the role of the transition prior $P(q_{t+1} \,|\, q_t) = \mathcal{N}(q_{t+1}, q_t, W^{-1})$ in the Bayesian view, with covariance matrix W^{-1}.
- The task constraints in classical motor planning can be interpreted as the *conditioning* of a task variable in the Bayesian view, which happens to be coupled (via the kinematics) to the actuator motions.
- Computing an optimal actuator motion in classical motor control corresponds to Bayesian *inference* of the posterior actuator motion conditioned on the task variables in the latter interpretation.

The example of redundant control is perhaps too minimalistic to appreciate the difference of the inference and the optimization view since, essentially, we only have one unconditioned random variable q_{t+1}. The Bayesian framework (which we call Bayesian Motor Control or BMC) produces a distribution over δq rather than only a single MAP solution – which is admittedly not crucial when looking only one step ahead. However, the full impact of this formulation becomes evident when considering whole motion trajectories, as we will see in the next section.

2.3 Planning as an Inference Problem

The previous sections considered the classical problem of inverse kinematics within a single time slice. Given the instantaneous desired motions in the task variables δx, we computed the instantaneous posterior motion in actuator space δq. However, in many circumstances, a preparatory movement in nullspace is advantageous to achieve future task constraints. One may call this a nullspace planning problem, where planning is used to determine the current nullspace movement that best allows to fulfill future task constraints.

It is straight-forward to extend the Bayesian control scheme to yield Bayes-optimal solutions to this trajectory planning problem: We now have a desired trajectory $x(t)$, rather than only instantaneous desired movement. Then, we compute the posterior over the full actuator trajectory $q(t)$, rather than only

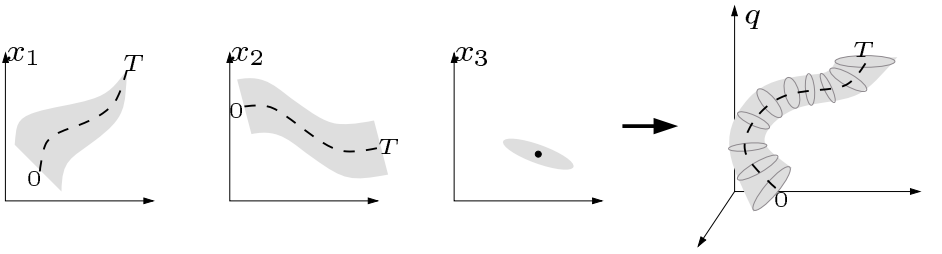
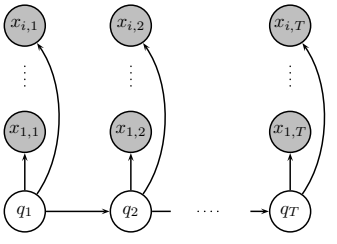


Fig. 1. Illustration of extended BMC: On $m = 3$ low-dimensional control variables x_i we have desired trajectories and associated tolerances. Here, x_3 is constrained to be kept in a constant range, x_1 needs to converge to a target. Extended BMC computes the posterior distribution over movements in posture space conditioned on these constraints.



fwd-bwd iterations k	$\int \lvert\dot{q}\rvert\ dt$	$E = \int \sum_i e_{i,t}\ dt$
$\frac{1}{2}$ (reactive controller)	13.0124	0.0873
1	7.73616	0.1366
$1\frac{1}{2}$	7.70018	0.0071
2	7.68302	0.0027
5	7.65795	0.0013
10	7.62888	0.0012

Fig. 2. (Fig) Dynamic Bayesian Network for extended BMC. (Table) Trajectory length and control errors after different number of forward-backward iterations of extended BMC. $k = \frac{1}{2}$ means a single forward pass and corresponds to the reactive forward controller using the single time slice BMC. $k = 1$ additionally includes a single backward smoothing.

over the instantaneous movement δq (refer Fig. 1). We formalize this in terms of a Dynamic Bayesian Network (DBN) where the computation of the posterior over $q(t)$ leads to a forward-backward inference procedure. Figure 2 shows the DBN for BMC under trajectory constraints. As for the single time slice case, we assume that

$$P(q_t \,|\, q_{t-1}) = \mathcal{N}(q_t, q_{t-1}, W^{-1}) \ , \quad P(x_{i,t} \,|\, q_t) = \mathcal{N}(x_{i,t}, \phi_i(q_t), \Sigma_i) \ . \quad (12)$$

We use Belief Propagation (BP) to describe the inference procedure (which could more classically be described as an iterative extended Kalman smoother). This simplifies the description of an iterative forward-backward, which yields more optimal results since the local linearizations of ϕ_i makes a single forward-backward only approximate. We represent the current belief over q_t as a Gaussian $\gamma_t(q_t) = \mathcal{N}(q_t, c_t, C_t)$. This belief is factored in three messages: the forward message α_t (from $q_{t-1} \rightarrow q_t$)

$$\alpha_t(q_t) = \int_{q_{t-1}} P(q_t \,|\, q_{t-1})\ \gamma_t(q_{t-1}) = \mathcal{N}(a_t, A_t)\ , \quad A_t = W^{-1} + C_{t-1}\ , \quad a_t = c_{t-1}, \tag{13}$$

the backward message β_t (from $q_{t+1} \ \rightarrow \ q_t$)

$$\beta_t(q_t) = \int_{q_{t+1}} P(q_{t+1} \,|\, q_t)\ \gamma_t(q_{t+1}) = \mathcal{N}(b_t, B_t)\ , \quad B_t = W^{-1} + C_{t+1}\ , \quad b_t = c_{t+1}, \tag{14}$$

and the observation message ϱ_t (from $x_{1:m,t} \ \rightarrow \ q_t$)

$$\varrho_t(q_t) = \prod_{i=1}^{m} P(x_{i,t} \,|\, q_t) = \mathcal{N}(R_t^{-1} r_t, R_t^{-1})\ , \tag{15}$$

$$r_t = R_t \hat{q} + \sum_i J_i^T \Sigma_i^{-1} (x_{i,t} - \phi_i(\hat{q}))\ , \quad R_t = \sum_i J_i^T \Sigma_i^{-1} J_i\ .$$

Here, we used a linearization of each ϕ_i at $\hat{q}$ (see below). We first initialize all γ's to be the uniform density (we use precision matrices), and $\gamma_1 = \mathcal{N}(q_0, \cdot 10^{-10})$. The inference procedure then iterates, updating

$$\gamma_t \leftarrow \alpha_t \varrho_t \beta_t \quad , \text{ i.e. }, \ C_t^{-1} \leftarrow A_t^{-1} + B_t^{-1} + R_t\ , \quad c_t \leftarrow C_t [A_t^{-1} a_t + B_t^{-1} b_t + r_t] \tag{16}$$

where α's, β's, and ϱ's are always recomputed using the above expressions, and linearizing at the mean $\hat{q} = c_t^{old}$ to compute the observation message (or $\hat{q} = a_{t-1}$ in the first forward iteration). The iterations are repeated alternating between forward for $t = 2, .., T$ and backward for $t = T{-}1, .., 2$. Next, we summarize some key properties of this framework:

1. The single time slice BMC is a special case of the extended BMC for $T = 2$ and if not iterating the belief propagation for $\gamma_{t=2}$. Iterating BP makes a difference since the point $\hat{q}$ of linearization of kinematics moves towards the final mean x_2 and thus, the result of BP converges to an exact step rather than the usual 1st order approximation.
2. Extended BMC allows us to continuously adjust the tightness Σ_i^{-1} of the control variables depending on time. For instance, we can impose that a desired control trajectory $x_i(t)$ is to be followed tightly in the early stage of the movement while only loosely in a later stage or vice versa.
3. Extended BMC solves a generalization of redundant movement planning: Redundant planning implies that a future goal is only determined by an endeffector constraint $x_T = x_T^*$ rather than a state space goal $q_T = q_T^*$. By adjusting the tightness (precision Σ^{-1}) of a control variable to be 0 for $t = 1, .., T - 1$ but tight in the last time step $t = T$, we can reproduce the classical redundant planning problem. Our approach generalizes this in that we can have an arbitrary number of control variables, and can adjust the control tightness e.g. to introduce intermediate goals or intermediate boundary conditions.
4. Even when we require tightness in the controls over the full trajectory, the extended BMC resolves the planning problem in the remaining nullspace.

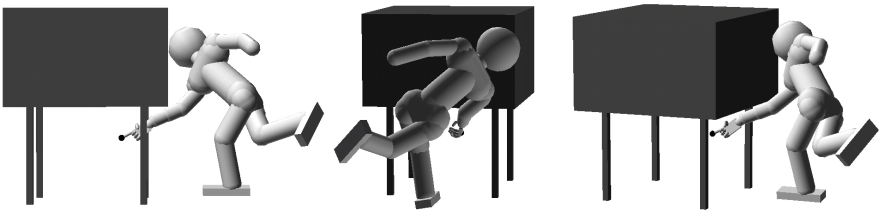


Fig. 3. Solution to a complex reaching movement under balance and collision constraints. The target for the right finger is illustrated as a black dot. Views from 3 different perspectives.

2.4 Examples for Extended BMC

We now consider the problem illustrated in Fig. 3 of reaching under an obstacle while keeping balance. This time the desired motion is not defined by reactive dynamical systems but rather by trajectories $x_{i,1:T}$ for each control variable x_i. We defined these to be linear interpolations from the start state to the target with $T = 100$, while keeping the precisions $\nu_{1:3} = (\cdot 10^3, \cdot 10^3, \cdot 10^6)$ constant over time. Figure 2(Table) displays the trajectory length and control errors after some forward-backward iterations. Note that the first line $k = \frac{1}{2}$ (only one forward pass) corresponds to the reactive application of the single time-slice BMC and thereby represents a classical forward controller. For instance, if we fix the total computational cost to 3 times that of a simple forward controller ($k = 1\frac{1}{2}$ iterations) we find an improvement of 40.8% w.r.t. the trajectory length and 91.9% w.r.t. control errors of extended BMC versus the forward controller. The reason for these improvements are that the forward controller chooses a non-efficient path which first moves straight towards the obstacle and later need longer motions to circumvent the obstacle. In contrast, the probabilistic smoothing of extended BMC leads to early nullspace movements (leaning to the right) which make the later circumvention of the obstacle more efficient.

2.5 Conclusion

Bayesian methods are often used for sensor processing and fusion. We investigated how Bayesian methods can also be applied on the motor level by abstracting the planning and constraint satisfaction as an optimal information processing paradigm. We proposed BMC to infer a single posterior body motion from multiple concurrent primitive controllers. Single time-slice BMC includes the standard and the prioritized inverse kinematics as a limit. However, we showed that BMC can lead to significant improvements in terms of trajectory length when compared to a strictly prioritized IK and can cope with singularity constraints (where $\text{rank}(J_i) < n_i$) and conflicting or infeasible constraints (where the nullspace projection N_i and thereby, the nullspace projected Jacobian $\bar{J}_i$ become singular). Further, BMC allows us to gradually change the required tightness of a constraint over time and to generate compromises between constraints. Extended BMC uses belief propagation to compute the posterior body trajectory conditioned on desired

control trajectories, and can be related to optimal control theory via Todorov's duality [4]. The total computational cost of a single backward sweep is equal to that of a single forward control sweep. We have demonstrated that this type of probabilistic smoothing can yield significant improvement in terms of the trajectory length and control errors over a forward controller. In effect, extended BMC generates early nullspace movements which make later motions more efficient. In this paper, we constrained ourselves to Gaussian belief representations, which also implies that there exists a dual quadratic cost minimization formulation. However, using existing inference techniques for other belief representations (e.g., particles, exponential families, or mixture of Gaussians), BMC can be extended to allow for a much wider range of constraint shapes.

BMC is one piece in a larger endeavor to apply Machine Learning techniques in the realm of robotics. Robotic control and behavior generation crucially depend on integrating information from as many sources as possible (sensors, prior knowledge, knowledge about constraints and future goals) to decide on current actions. The Bayesian framework is an ideal candidate for this. Our goal is to understand the problem of behavior and motion generation as a single coherent inference process in a structured model containing many sources of information and constraints. By integrating BMC in a larger framework such as Grupen's control paradigm [5] and combining it with more complex models of motor primitives [6,7,8], we will advance towards a complete, coherent Bayesian framework for behavior and motion generation.

3 Learning Dynamics under Multiple Contexts

In the previous section, we argued that principle problems of movement generation or planning can be framed as information processing problems, given information about goals, targets or constraints, and *assuming knowledge about the dependency of random variables.* This relationship between variables of interest is often the dynamics of the plant or the robot; a relation that defines the state transitions for various actuation torques.

In many situations, exact analytical derivation of the robot dynamics is not feasible. This can be either due to the complexity of the system or due to lack of or inaccurate knowledge of the physical properties of the robot. Moreover, the dynamics of the robot often depend on a varying unobserved external context and exhibit non-stationarity. An example of unobserved external context that results in non-stationary dynamics is the work load of a manipulator: the resultant dynamics of the robot arm change as it manipulates objects with different physical properties, e.g. mass or mass distribution. Adaptive control and learning methods can be used in cases of non-stationary dynamics. However, if the dynamics switch back and forth, e.g. if manipulating a set of tools for executing various tasks, classic adaptive control methods are inadequate since they unlearn the dynamics of the previously experienced contexts and relearn them when they reoccur. Furthermore, there may be large errors and instability during the period of readaptation.

Fig. 4. Graphical model representing the (a) forward and (b) inverse model

In the next section, we will look at methods from the domain of statistical machine learning to efficiently acquire dynamics from movement data, which are then extended to representing, learning and switching between multiple models, each of which is appropriate for a different context.

3.1 Statistical Learning of Forward and Inverse Dynamic Models

Anthropomorphic robotic systems have complex kinematic and dynamic structure, significant non-linearities and hard to model non-rigid body dynamics; hence, deriving reliable analytical models of their dynamics can be cumbersome and/or inaccurate. We take the approach of learning dynamics for control from movement data.

At time step t, let $\Theta_t = (q_t, \dot{q}_t)$ be the state of the system (which includes the position and velocity components) and τ_t the control signal. A deterministic forward model f describes the discrete-time system dynamics as

$$\Theta_{t+1} = f(\Theta_t, \tau_t) \ . \tag{17}$$

Learning a forward model f of the dynamics is useful for predicting the behavior of the system. However, for control purposes, an inverse model is needed. The inverse model g maps from transitions between states to the control signal that is needed to achieve this transition:

$$\tau_t = g(\Theta_t, \Theta_{t+1}) \ . \tag{18}$$

A probabilistic graphical model representation of the forward and inverse model is shown in Fig. 4(left) and Fig. 4(right), respectively.

The inverse model shown in Fig. 4(right) can be used in many control settings; one of the most common being to use it as part of a composite controller. Given a desired trajectory, $\Theta^*_{1:T}$, the composite control law computes the command as

$$\tau_t = g(\Theta^*_t, \Theta^*_{t+1}) + K_p \ (q^*_t - q_t) + K_u \ (\dot{q}^*_t - \dot{q}_t) \ , \tag{19}$$

where K_p and K_u are gain matrices. This is a combination of a feedforward command that uses the inverse model and a feedback command that takes into account the actual state of the system. The more accurate the inverse model is,

(a) (b)

Fig. 5. Robotic platforms: (a) SARCOS dextrous arm (b) DLR LWR III arm

the lower the feedback component of the command will be, i.e., the magnitude of the feedback command can be used as a measure of the accuracy of the inverse model. Furthermore, good predictive models allow us to use low feedback gains, resulting in a highly compliant system without sacrificing the speed and accuracy of the movements.

Typically, in robotic systems with proprioceptive and torque sensing, at each time step t we "observe" a state transition and an applied torque signal summarized in the triplet $(\Theta_t, \Theta_{t+1}, \tau_t)$, i.e., we have access to the true applied control command (which was generated via composite control). To learn the inverse dynamics, we need a *non-linear, online* regression technique. We use Locally Weighted Projection Regression (LWPR) [9] – an algorithm which is extremely robust and efficient for incremental learning of non-linear models in high dimensions. An LWPR model uses a set of linear models, each of which is accompanied by a locality kernel (usually a gaussian) that defines the area of validity of the linear model. For an input $\boldsymbol{x}$, if the output of the k^{th} local model is written as $y_k(\boldsymbol{x})$ and the locality kernel activation is $w_k(\boldsymbol{x})$, the combined prediction of the LWPR model, $\hat{y}$, is

$$\hat{y}(\boldsymbol{x}) = \frac{1}{W} \sum_k w_k(\boldsymbol{x})\ y_k(\boldsymbol{x}) \ , \quad W = \sum_k w_k(\boldsymbol{x}) \ . \tag{20}$$

The parameters of the local linear models and locality kernels are adapted online and also local models are added on an as needed basis. Furthermore, LWPR provides statistically sound input dependent confidence bounds on its predictions and employs Partial Least Squares (PLS) to deal with high dimensional inputs. For more details about LWPR, see [9] and for an efficient implementation, refer to [10].

The role of LWPR in the probabilistic inverse model of Fig. 4 can be summarized in the equation:

$$P(\tau \mid \Theta_{t+1}, \Theta_t) = \mathcal{N}(\phi(\Theta_{t+1}, \Theta_t),\ \sigma(\Theta_{t+1}, \Theta_t)), \tag{21}$$

whose $\phi(\Theta_{t+1}, \Theta_t)$ is a learned LWPR regression mapping state transitions to torques. Here, we have two options for choosing the variance: (1) we can assume a fixed noise level independent of the context and the input, e.g. a maximum likelihood estimate; (2) we can use the confidence bounds provided by each LWPR model which also depends on the current input (Θ_{t+1}, Θ_t), this will give higher noise levels in areas where not much data has been seen. We will test both cases in our experiments. Please see [9] for more details on LWPR and the input dependent variance estimate.

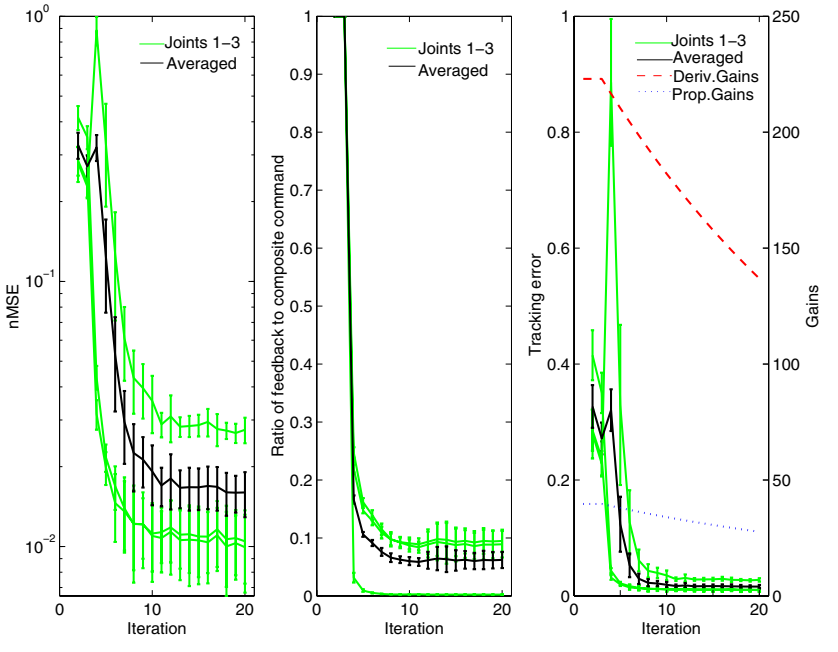


Fig. 6. Results on learning stationary dynamics on a 3 DOF simulated robot arm. Left: test error. Middle: contribution of error-correcting feedback command. Right: Tracking error.

Experiments in Learning Dynamics for Single Context. We verify the ability to learn the inverse model online with LWPR and show that the model can successfully be used for control. We demonstrated this for a simulated 3 DOF robot arm[1] as well as on the 7 DOF anthropomorphic SARCOS robot arm (Fig. 5(a))and recently, on the DLR LWR arm (Fig. 5(b)). Here, the statistics are accumulated and shown briefly for the simulated arm. The task of the arm was to follow a simple trajectory planned in joint angle space, consisting of a superposition of sinusoids with different phase shifts for each joint:

[1] Simulations performed using ODE and OpenGL.

$$\theta_i^* = a_i \cos(\alpha_i \frac{2\pi}{T} t) + b_i \cos(\beta_i \frac{2\pi}{T} t) \ , \tag{22}$$

where $T = 4000$ is the total length of the target trajectory, $a_i, b_i \in [-1, 1]$ are different amplitudes and $\alpha_i, \beta_i \in \{1, .., 15\}$ parameterize different frequencies. 20 iterations of the trajectory were repeated: during the first four iterations, pure feedback (PD) control was used to control the arm, while at the next 16 iterations, a composite controller using the inverse model being learned was used. The gains were lowered as training proceeded. The procedure was executed ten times, for ten different contexts, for accumulating statistics. Different contexts are simulated by attaching an object with different mass at the last link of the arm.

Figure 6(left) plots the normalized mean squared error between the torques predicted by the LWPR model and the true torques experienced on the test data (i.e., the data that was held out from the training), which shows a quick drop as training proceeds and settles at a very low value averaged over all trials. The contribution of the error-correcting feedback command to the feedforward command (see Fig. 6(middle)) is low, vouching for the accuracy of the learnt model while being used for control. Furthermore, the tracking error (Fig. 6(right)) is very low and improves significantly when we switch to composite control. For the detailed statistics on the online dynamics learning of the 7 DOF SARCOS robot arm and tracking results on a pattern eight task, readers are referred to [9].

3.2 Inference of a Discrete Latent Context Variable

The multiple model paradigm copes with the issue of non-stationary dynamics by using a set of models, each of which is specialized to a different context. A schematic of a generic multiple model paradigm is shown in Fig. 7. The observed dynamics of the system are compared to the prediction of each learned model

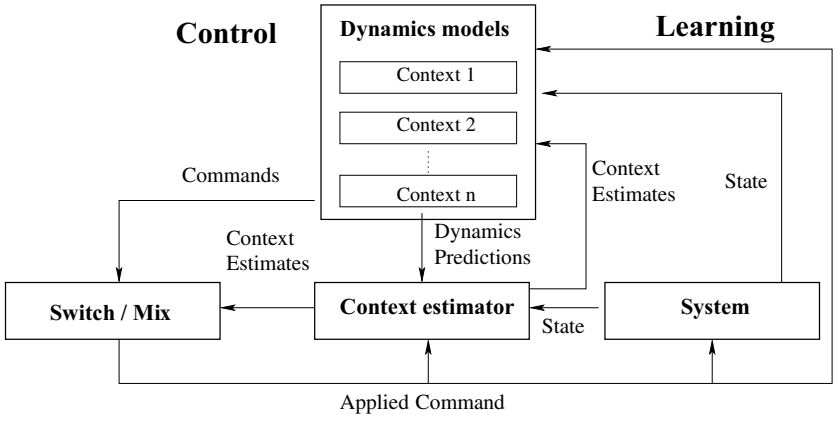


Fig. 7. Schematic of a multiple model paradigm

to identify the current context. The context estimates are used for selecting the model to use for control and for training. All existing multiple model paradigms roughly follow the same plot. The main issues that have to be tackled for using multiple discrete models for control are:

1. Infer the current context for selecting the appropriate model to use for *control*.
2. Infer the current context for selecting the appropriate model to *train* with the experienced data.
3. Estimate the appropriate *number of models* (possibly using a novelty detection mechanism).

Context Estimation. It is appropriate to formulate context estimation in a probabilistic setting to account for inaccuracies of the learnt models as well as handle transitions. Apart from dealing with uncertainty and context estimation in a principled way, the probabilistic formulation can be useful for novelty detection. That is, if experienced data is not likely for any of the learned models, it can be classified as novel and used to train a new model.

While in most multiple model approaches, context estimation is performed by comparing the predictions of a forward model to the dynamic behaviour of the system, in the absence of redundant actuators, one can use the inverse model as well – an approach we will follow here. The graphical model in Fig. 8(a) represents a set of inverse models corresponding to a specific number of contexts. The hidden contextual variable c_t is discrete and indexes the different models.

The inverse model in this formulation can be written as:

$$P(\tau \mid \Theta_{t+1}, \Theta_t, c_t = i) = \mathcal{N}(\phi^{(i)}(\Theta_{t+1}, \Theta_t),\ \sigma^{(i)}(\Theta_{t+1}, \Theta_t)) \ , \tag{23}$$

where $\phi^{(i)}$ is the command predicted by the LWPR model corresponding to the i^{th} context and $\sigma^{(i)}$ is some estimate of the variance, which again can be either set to a predetermined constant or based upon the input dependent confidence bounds provided by LWPR. Also, we can either assume knowledge of the prior probability of contexts or we can assume that different contexts have equal prior probabilities $P(c_t)$. Under this probabilistic formulation, context estimation is just inferring the posterior of c_t given a state transition and the command that resulted in this transition:

$$P(c_t = i \mid \Theta_t, \Theta_{t+1}, \tau_t) \propto P(\tau_t \mid \Theta_t, \Theta_{t+1}, c_t = i)\ P(c_t = i). \tag{24}$$

Context estimates are very sensitive to the accuracy of the inverse models. They can be improved by acknowledging that contexts do not change too frequently. We can introduce a temporal dependency between contexts $P(c_{t+1} \mid c_t)$ with an appropriate transition probability between contexts that reflects our prior belief of the switching frequency to achieve much more robust context estimation. The graphical model can be reformulated as the Dynamic Bayesian Network shown in Fig. 8(b) to achieve this. Application of standard Hidden Markov Model (HMM) techniques is straightforward by using (24) as the observation likelihood in the HMM, given the hidden state $c_t = i$. A low transition probability penalizes too frequent transitions and using filtering, smoothing or Viterbi alignment produces more stable context estimates.

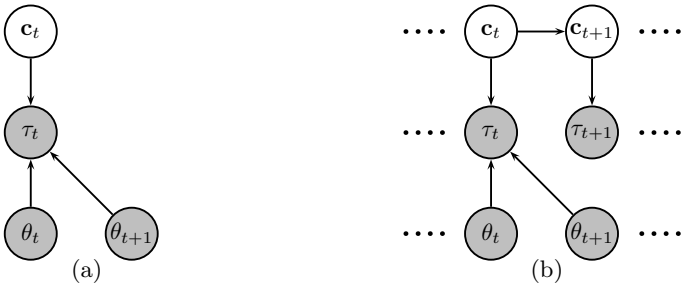


Fig. 8. Multiple models and hidden contexts: (a) Graphical representation of hidden latent context within the dynamics (b) DBN to capture the temporal dependencies between the latent contexts

Data Separation for Learning. The problem of bootstrapping the context separation from context-unlabeled data is very similar to clustering problems using a mixture of Gaussians. In fact, the context variable can be interpreted as a latent mixture indicator and each inverse model contributes a mixture component to give rise to the mixture model of the form $P(\tau_t \,|\, \Theta_t, \Theta_{t+1}) = \sum_i P(\tau_t \,|\, \Theta_t, \Theta_{t+1}, c_t = i)\ P(c_t = i)$. Clustering with mixtures of Gaussians is usually trained using Expectation-Maximization (EM), where initially the data are labeled with random responsibilities (are assigned randomly to the different mixture components). Then every mixture component is trained on its assigned (weighted) data (M-step) and afterwards the responsibilities for each data point is recomputed by setting them proportional to the likelihoods for each mixture component (E-step). Iterating this procedure, each mixture component specializes on different parts of the data and the responsibilities encode the learned cluster assignments.

We apply the EM algorithm for separating the data and learning the models. In our case, the likelihood of a data triplet $(\Theta_t, \Theta_{t+1}, \tau_t)$ under the i^{th} inverse model is $P(\tau_t \,|\, \Theta_t, \Theta_{t+1}, c_t = i)$, which is a Gaussian that could have either fixed variance or variance given by LWPR's confidence bounds. Learning the transition probabilities from a sequence of observations is straightforward using EM. In particular, the probabilities $p(c_t, c_{t+1} \,|\, \Theta_{1\ldots T})$ for $t = 1 \ldots T-1$ need to be calculated (E-step), a straightforward problem in HMM inference and from these, the relative frequencies $p(c_{t+1} \,|\, c_t)$ for any t can be easily estimated (M-step). As usual, the procedure is iterated a few times: i.e. $p(c_t, c_{t+1} \,|\, \Theta_T)$ is computed using some values for the transition probabilities (one could initially set all transitions to be equally probable), then $p(c_{t+1} \,|\, c_t)$ is estimated, then $p(c_t, c_{t+1} \,|\, \Theta_{1\ldots T})$ is computed again using the estimated transition probabilities and so on until either a maximum number of iterations is reached or some other criterion is met, e.g. the likelihood of the observed data stops increasing. We will see examples of this procedure in the next section.

Experiments With Multiple Discrete Models. The context estimation, transition probability learning and separation of experience methods suggested

in the previous sections were tested on the simulated arm. Here different contexts are simulated by varying the mass on the last link of the manipulator.

Random switches between six contexts were performed in the simulation, where at every time step we switch to a random context with probability .001 and stay in the current context otherwise.

We have two classes of experiments, one is where we are not using HMM filtering of the contextual variable and the other is where we use it. Also, we have two choices for the variance of the observation model, one is where we use a constant (set at the MSE on the test data) and the other is where we use the more principled confidence bounds provided by LWPR. For the moment we assume that for the temporal model, we know the correct transition probabilities. The simulation was run for 10 iterations. We run 5 different runs of the simulation, switching between six different contexts each time and we start by examining the accuracy of our probabilistic context estimation methods while not using the context estimates for control (a PD controller was used). The percentage of accurate online context estimates for the four cases, averaged over the five trials, can be seen in Fig. 9(a)(error bars are obtained from the five different trials).

Figure 9(b) gives an example of how the best context estimation method that we have, the HMM filtering using LWPR's confidence bounds, performs when used for online context estimation and control. Sometimes the context estimation lags behind a few time steps when there are context switches, which is a natural effect of online filtering (as opposed to retrospect smoothing).

The context estimates were then used online for selecting the model that will provide the feed-forward commands. Figure 9(c) shows the percentage of accurate online context estimates for the four case. Results are again averaged over the five trials. The performance of online context estimation and control is close to the control performance we achieved for the single context displayed in Fig. 6.

Furthermore, we investigate the bootstrapping of data separation / model learning. We will use the temporal model as it has been shown to give far superior results for context estimation and control and thus we also investigate learning of transition probabilities. Again, when generating the data, we switched between two different contexts with probability .001 at each time step, however we now do not use the correct transition probabilities in either inference (E-step) or learning (M-step). We first collected a batch of context-unlabeled data from 5 cycles through the target trajectory where the arm was controlled by pure feedback PD control. The EM procedure for data separation and learning of transition probabilities (Sec. 3.2) was applied. As mentioned before, using the confidence bounds for the noise estimates of the observation (inverse) model from the beginning of the EM procedure does not usually work and makes all models collapse into the same model. Using the maximum likelihood estimates works much better, however, this still does not give very good results. The problem lies in the fact that a local learning method is used: data seem to be separated correctly locally but not globally, across the input space. This is demonstrated in Fig. 10(a). A possibility would be to try to regroup the local models to maximize the smoothness of the learned model, however, we show here that it is possible to

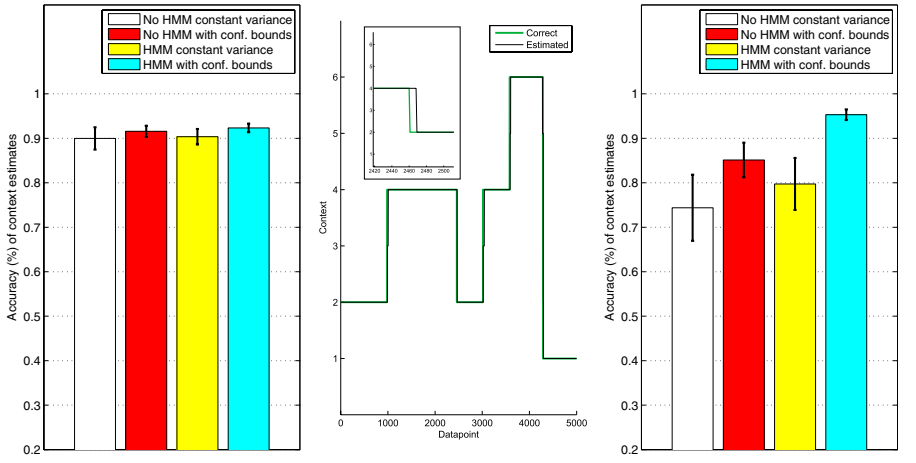


Fig. 9. Online context estimation without using the context estimates for control. (a) Context estimation accuracy using different estimation methods. (b) Example of random context switches and its estimate using HMM filtering over time. (c) Online context estimation using the context estimates for control.

achieve perfect data separation by modifying the EM algorithm slightly. First, we note that if we manage to increase the noise only on the areas where there is mixing between local models that actually belong in different contexts, the posterior of the datapoints in that area will switch slower and less frequently in that region. This increase of noise can be achieved using LWPR's confidence bounds: the confidence bounds increase when there is a sudden change in the model's prediction. Thus, we run the EM procedure using a maximum likelihood estimate for the inverse model noise until the data is well separated locally and then switch to using the confidence bounds. This trick has been sufficient to solve the problem in some cases but not always. If at the point that we start to use the confidence bounds, there are very large or too many areas that need to be swapped between LWPR models, then the procedure may still get stuck. The way to solve this problem is to also switch from using smoothed estimates in the E-step, to using filtered estimates. This effectively, together with the increased noise levels on edges of the areas that need to be swapped, makes the areas that are not grouped correctly narrower and narrower in each iteration of EM. This modified EM procedure was tried with perfect data separation always being achieved. Figure 10(b) displays a typical evolution of the data separation. Switching to using the confidence bounds and the filtered instead of the smoothed estimates happens at iteration 20.

The transition probabilities are also estimated during the EM procedure: the estimated probability are very close to the actual ones, i.e. 0.999 staying in the same context and 0.001 switching.

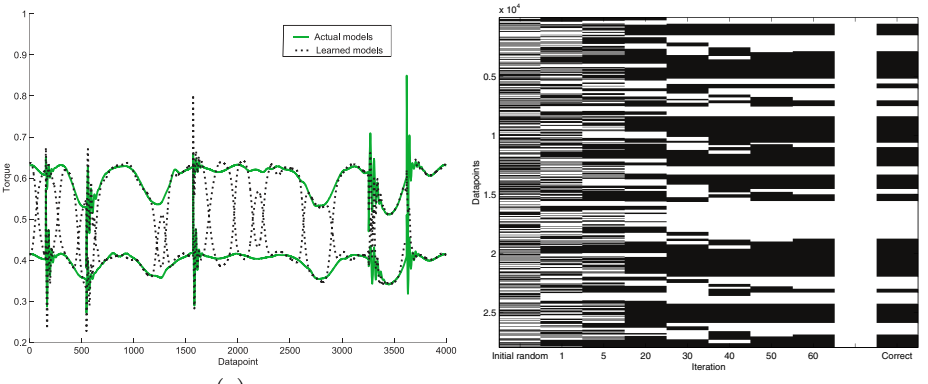


Fig. 10. (a) The solid lines show the predictions of the inverse models for the first joint on the training data if the models had been trained with perfectly separated data. The dotted lines show the predictions of the models generated by the automated separation procedure. Data separation seems to work locally but not globally. (b) The evolution of the data separation from unlabeled data over some iterations of the EM-procedure. The first column displays the initial random assignment of datapoints to contexts. The last column displays the correct context for each datapoint. The columns in between display the most likely context for each datapoint according to the currently learned models for some iterations of the EM. procedure.

3.3 Augmented Model for Continuous Contexts

The multiple model paradigm has several limitations. First of all, the right number of models needs to be known or estimated. Estimating the number of contexts only from data (using some model selection procedure) is a non-trivial problem. Realistically, novel contexts appear quite often and to cope with this, a novelty detection mechanism is needed. However, even with a very robust novelty detection mechanism, we may end up with a very large number of models, since for most scenarios, possible contexts are infinite. Moreover, we would like to generalize between contexts and most multiple model paradigms do not cope well with this.

All these issues can be circumvented if the set of models is replaced with a single model that takes as additional input appropriate *continuous* hidden contextual variables, i.e., instead of a set of g_is corresponding to different contexts, a single inverse model G is used:

$$\tau_t = G(\Theta_t, \Theta_{t+1}, c_t) \ . \tag{25}$$

Here, c_t is not a discrete variable that indexes different models but a set of continuous variables that provides information about the context. The probabilistic model of the inverse dynamics would then be:

$$P(\tau \mid \Theta_t, \Theta_{t+1}, c_t) = \mathcal{N}(G(\Theta_t, \Theta_{t+1}, c_t),\ \sigma(\Theta_t, \Theta_{t+1}, c_t)) \ . \tag{26}$$

A possibility for learning the augmented model is to follow the same procedure as in the discrete case for learning the models, i.e., apply an EM like procedure. Using the same temporal dependency formalization, this results in a state space model. Learning in nonlinear state space models is discussed in [11], [12] and [13]. However, the relationship of the contextual variables to the output of the augmented model could be arbitrary, making learning in such a setting a very difficult task.

It is imperative to exploit any prior knowledge about the relationship of the inverse model to appropriate contextual variables. For the case of manipulation of objects with a robot arm (see schematic of the robot arm link with load in Fig. 11), we can take advantage of the fact that the dynamics of a robot arm have a linear relationship to the inertial properties of the links. In other words, the inverse dynamics can be written in the form:

$$\tau = Y(q, \dot{q}, \ddot{q})\pi \tag{27}$$

where q, $\dot{q}$ and $\ddot{q}$ denote joint angles, velocities and accelerations respectively. This relationship can be derived based on fundamentals of robot dynamics [14,15] as shown in Table 1. This equation splits the dynamics in two terms. Say the manipulator has n joints, then $Y(q, \dot{q}, \ddot{q})$ is a $n \times 10n$ matrix that depends on kinematics properties of the arm such as link lengths, direction of axis of rotation of joints and so on. This is a complicated and non-linear function of joint angles, velocities and accelerations. The term π is a $10n$-dimensional vector containing the inertial parameters of all links of the arm (see Table 1).

Now, let's examine how this can be used to acquire the augmented model for the scenario of changing loads. The important thing to note is that the kinematics dependent term Y does not change as different objects are manipulated. Only the inertial parameters of the last link of the arm change, i.e. the last 10 elements of the vector π.

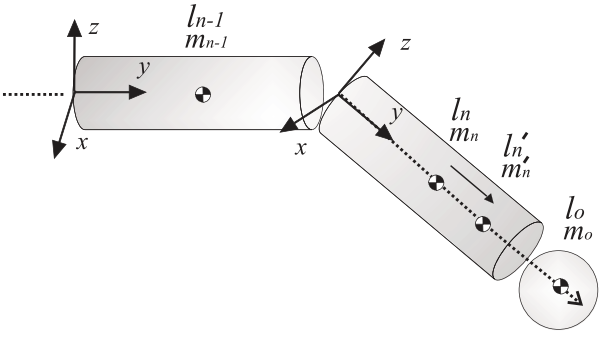


Fig. 11. Schematic of the load and inertial parameters involved in manipulator dynamics

Table 1. Linearity of the dynamics model in the inertial parameters

If $\mathcal{T}$ is the kinetic energy, $\mathcal{U}$ is the potential energy of the system and we define a Lagrangian $\mathcal{L} = \mathcal{T} - \mathcal{U}$, the dynamics of the system is given by

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}_i} - \frac{\partial \mathcal{L}}{\partial q_i} = \tau_i \tag{28}$$

where $q_1, q_2 ... q_n$ is a set of generalized coordinates (here, the joint angles) and $\tau_1, \tau_2 ... \tau_n$ denote the so called generalized forces associated with the corresponding joint angles q_i. The generalized force τ_i is the sum of joint actuator torques, joint friction torques and other forces acting on the joint (e.g. forces induced by contact with the environment). The total kinetic energy $\mathcal{T}$ and the total potential energy $\mathcal{U}$ is just the sum of the kinetic energy and potential energies of all the links of the manipulator respectively, i.e., $\mathcal{T} = \sum_{j=1}^{n} \mathcal{T}_j$, $\mathcal{U} = \sum_{j=1}^{n} \mathcal{U}_j$ The kinetic and potential energy of the j^{th} link is given by:

$$\mathcal{T}_j = \frac{1}{2} m_j \dot{p}_j^T \dot{p}_j + m_j l_j \dot{p}_j^T S(\omega_j) + \frac{1}{2} \omega_j^T I_j \omega_j \quad , \quad \mathcal{U}_{l_i} = -m_j g_0^T p_j - m_j g_0^T l_j \tag{29}$$

where m_j is the total **mass of link** j, p_j is the position vector of the origin of frame j expressed in the base frame, ω_j is the rotational velocity of link j, $S(\omega_j)$ is a 3×3 skew-symmetric matrix that depends on ω_j, l_j is the position vector of the center of mass of the link from the origin of the frame of the link, g_0 is the gravity acceleration vector, I_j is the **inertia tensor of link** j measured at the origin of the reference frame of the link. Substituting (29) in the Lagrangian and with some rearrangement, we can see that the Lagrangian has a linear relationship to the set of inertial parameters:

$$\pi = [m_1, m_1 l_{1x}, m_1 l_{1y}, m_1 l_{1z}, I_{1xx}, I_{1xy}, ..., m_n, m_n l_{nx}, m_n l_{ny}, m_n l_{nz}, I_{nxx}, ..., I_{nzz}] \tag{30}$$

In short, the Lagrangian can be written in the form:

$$\mathcal{L} = g(q, \dot{q})\pi \tag{31}$$

Since the inertial parameters in π do not depend on time or $\dot{q}$ then the dynamics equation for joint i is:

$$\pi \frac{d}{dt}\frac{\partial g(q, \dot{q})}{\partial \dot{q}_i} - \pi \frac{\partial g(q, \dot{q})}{\partial q_i} = \tau_i \tag{32}$$

Thus, the dynamics can be written in the form

$$\tau_i = y_i(q, \dot{q}, \ddot{q})\pi \tag{33}$$

It is worth noting that we didn't take into account the dynamics of the motor attached to each link. In that case, another element per link is added to the vector π, giving a total of 11 inertial parameters per joint. For more details see [14]. We will ignore the motor dynamics, however, our general arguments can easily be extended in order to include the dynamics of the motor.

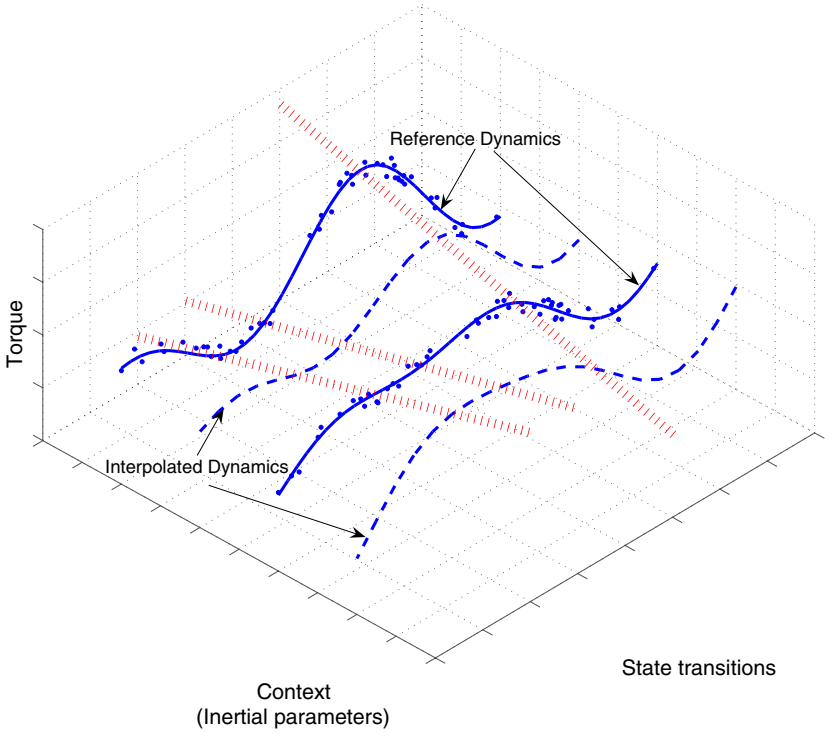


Fig. 12. Learning the augmented model. The dots are training data for the different contexts. The solid lines are the learned models for each context, the red dotted lines show the interpolation of the augmented model predictions from a set of learned models and the dashed lines show the global augmented model for some new context.

Denoting the 10 inertial parameters of the union of the last link and manipulated object as π_o and using these as the contextual variables, the augmented model $G(\Theta_t, \Theta_{t+1}, c_t)$ can be written as:

$$\tau_t = G(\Theta_t, \Theta_{t+1}, c_t) = A(q, \dot{q}, \ddot{q}) + B(q, \dot{q}, \ddot{q})\pi_o \tag{34}$$

Here, the matrix $B(q, \dot{q}, \ddot{q})$ consists of the last 10 columns of the matrix $Y(q, \dot{q}, \ddot{q})$ and $A(q, \dot{q}, \ddot{q})$ is the $n-$dimensional vector given by multiplying the array consisting of the first $(n-1) \times 10$ columns of $Y(q, \dot{q}, \ddot{q})$ with the vector consisting of the inertial parameters of the first $n-1$ links. Note that state transitions have been appropriately replaced by joint angles, velocities and accelerations. This is more compactly written as:

$$G(\Theta_t, \Theta_{t+1}, c_t) = \tilde{Y}(q, \dot{q}, \ddot{q})\tilde{\pi_o} = \tau \tag{35}$$

where $\tilde{\pi_o}$ denotes the vector $[1\ \pi_o^T]^T$ and $\tilde{Y}(q, \dot{q}, \ddot{q})$ denotes the $n \times 11$ matrix $[A(q, \dot{q}, \ddot{q})\ B(q, \dot{q}, \ddot{q})]$.

To *acquire* the model, essentially means to estimate $\tilde{Y}(q, \dot{q}, \ddot{q})$. If we have an appropriate number of learned models (that is, at least as many as the cardinality of $\tilde{\pi_o}$) and the corresponding π_o labels, we can simply estimate $\tilde{Y}(q, \dot{q}, \ddot{q})$ using least squares due to the linearity property. Say, we have learned a set of *reference* models $g^1(q, \dot{q}, \ddot{q}), g^2(q, \dot{q}, \ddot{q})...g^l(q, \dot{q}, \ddot{q})$ corresponding to manipulation of objects which result in the last link of the arm having known inertial parameters $\pi_o^1, \pi_o^2...\pi_o^l$, one can just evaluate $\tilde{Y}(q, \dot{q}, \ddot{q})$ as:

$$\tilde{Y}(q, \dot{q}, \ddot{q}) = T(q, \dot{q}, \ddot{q})\tilde{\Pi}^T(\tilde{\Pi}\tilde{\Pi}^T)^{-1} \tag{36}$$

Where, $T(q, \dot{q}, \ddot{q})$ is a matrix with the reference models' predictions as its columns and $\tilde{\Pi}$ is a matrix with the reference models' inertial parameters.

The augmented model can be used both for control and context estimation purposes. For *control* purposes, say we have an estimate of π_o at time t, given the desired transition for the next time step, we can easily compute $\tilde{Y}(q^*, \dot{q^*}, \ddot{q^*})$ and hence, the feedforward command. For robust context estimation, we can use temporal dependencies, similar to the principles used in the multiple model scenario. However, since we now have a set of continuous hidden variables as opposed to a single discrete context variable, the inference is slightly more involved (refer to Table 2).

A schematic for acquisition of the augmented model is displayed in Fig. 12. The dots are data belonging to different contexts. A model is fit to the data belonging to each of the contexts (the solid lines) and then, we can use the predictions of the learned models together with the known corresponding inertial parameters to do a least squares estimate and acquire the augmented inverse model for any point of the input space (the dotted lines). Computing the augmented model for any point of the input space gives the dynamics model of any other context (dashed lines).

It is important to note that to acquire the augmented inverse model, the regression coefficient matrix $\tilde{Y}(q, \dot{q}, \ddot{q})$ has to be evaluated at all relevant points in the input space. However, this is not as computationally expensive as it might seem at first. All that is needed is to reevaluate the predictions of the reference inverse models at each point in input space and multiply by the pseudoinverse of the reference inertial parameters matrix $\tilde{\Pi}^T(\tilde{\Pi}\tilde{\Pi}^T)^{-1}$. This pseudoinverse needs to be evaluated once and no further matrix inversion is needed to solve the inverse problem at all points in the input space.

The previous discussion implies that, ideally, if we have the prerequisite number of 'labeled' context models (at least eleven independent models), then, one can deal with manipulation of any object. In practice, however, since learned dynamic models will not be perfect and due to the presence of noise in the sensor measurements, a larger number of 'context models' may be necessary to give accurate estimates and control.

Experiments with the Augmented Model. The augmented model proposed for extracting the continuous context/latent variable was empirically evaluated.

In our experiments, both the center of mass of the last link l_n and the load l_o are constrained to lie on the y axis of the last link's reference frame, so that

Table 2. Inferring the hidden continuous context in the temporal model

In our probabilistic setting, the augmented inverse model is

$$\tau_t = G(\Theta_t, \Theta_{t+1}, c_t) = A(\Theta_t, \Theta_{t+1}) + B(\Theta_t, \Theta_{t+1})c_t + \eta \tag{37}$$

where $A(\Theta_t, \Theta_{t+1})$ and $B(\Theta_t, \Theta_{t+1})$ are estimated from the models used for forming the augmented model and $\eta = \mathcal{N}(0, \Sigma_{obs})$. Σ_{obs} is estimated from the confidence bounds of the inverse models that form the augmented model. Also, the transition model for the context needs to be defined. Since we believe that the context does not change too often, this is set to:

$$c_{t+1} = c_t + \zeta \tag{38}$$

where $\zeta = \mathcal{N}(0, \Sigma_{tr})$ with Σ_{tr} set to a very small value.
Based on the defined model, we can write down the inference for the temporal Bayesian network using the augmented inverse model. For control, only filtered estimates (a la Kalman filtering) can be used.
We want to compute $p(c_t \,|\, \tau_{1:t+1}, \Theta_{1:t+1})$ using the estimate at the previous time step $p(c_{t-1} \,|\, \tau_{1:t}, \Theta_{1:t})$ and the new evidence τ_{t+1} and Θ_{t+1}. The previous estimate $p(c_{t-1} \,|\, \tau_{1:t}, \Theta_{1:t})$ is defined as:

$$p(c_{t-1} \,|\, \tau_{1:t}, \Theta_{1:t}) = \mathcal{N}(\mu_{t-1\,|\,t}, \Sigma_{t-1\,|\,t}) \tag{39}$$

Estimates for the next time step $p(c_t \,|\, \tau_{1:t+1}, \Theta_{1:t+1})$ are obtained in a recursive way in two steps. The first is the **prediction** step where, $p(c_t \,|\, \tau_{1:t}, \Theta_{1:t})$ is computed using the filtered estimate on the previous time step and the transition model $p(c_{t+1} \,|\, c_t)$, without taking into account evidence at time $t + 1$:

$$p(c_t \,|\, \tau_{1:t}, \Theta_{1:t}) = \mathcal{N}(\mu_{t\,|\,t}, \Sigma_{t\,|\,t}) \tag{40}$$

where $\mu_{t\,|\,t} = \mu_{t-1\,|\,t}$ and $\Sigma t \,|\, t = \Sigma_{t\,|\,t} + \Sigma_{tr}$. Then, the **filtered estimate** modifies the predicted estimates using the observation at the time $t + 1$ as (dependency of A and B on the state transition is omitted for compactness):

$$p(c_t \,|\, \tau_{1:t+1}, \Theta_{1:t+1}) = \mathcal{N}(\mu_{t\,|\,t+1}, \Sigma_{t\,|\,t+1}) \tag{41}$$

where,

$$\mu_{t\,|\,t+1} = \mu_{t\,|\,t} + \Sigma_{t\,|\,t} B^T (B\Sigma_{t\,|\,t} B^T + \Sigma_{obs})^{-1} (\tau_{t+1} - A - B\mu_{t\,|\,t}) \tag{42}$$

$$\Sigma_{t\,|\,t+1} = \Sigma_{t\,|\,t} - \Sigma_{t\,|\,t} B^T (B\Sigma_{t\,|\,t} B^T + \Sigma_{obs})^{-1} B\Sigma_{t\,|\,t} \tag{43}$$

the center of mass of their union $\acute{l}_n$ also lies on the y axis (refer Fig. 11). The problem was constrained in a way that, for the three degrees of freedom robot arm of our experiments, only three out of the ten inertial parameters of the last link could be estimated and thus, three contextual variables were needed to describe the augmented model. These inertial parameters are the mass, the mass $\times$ the $y - position$ of the center of mass and the moment of inertia around the z axis . This was achieved by choosing the coordinate system attached to the last link such that the center of mass both of the link and the object lie on the y axis

(see Fig. 11). Thus, mass $\times$ the $x - position$ and mass $\times$ the $z - position$ are zero. Furthermore, the off-diagonal elements of the inertia tensor are zero and only the moment of inertia around the z axis has significant contribution to the dynamics.

Unlike before, now both the mass and shape of the manipulated object change randomly and can take any value in a specific range. Again, we start by not using the context estimates for control, i.e. we apply PD control. We then repeated the same experiments but using the context estimates for control to see if the accuracy of our continuous context estimates is sufficient for motor control. Experiments were executed for both cases, five times, using different reference models for each of the runs. Figure 13(a) shows the estimation accuracy of the three context variables for the no control / control cases. The error measure used is the nMSE on the target variable. We can see that the relative accuracy is not significantly dfferent. Figure 13(b-d) show a snapshot of the actual and estimated contextual variables. The mass and y-position of the center of mass $\times$ the mass were more accurately estimated than the moment of inertia around the y axis. For the case that the context estimates were used for control, the average ratio of feedback to composite command was 0.1428 with standard deviation between the runs 0.0141.

3.4 Discussion

We have described a method of using a learned set of models for control of a system with non-linear dynamics under *continuously* varying contexts. In addition , we have refined the multiple model paradigm to be able to *simultaneously* deal with learning dynamic models, use them for online switching control and also efficiently bootstrap data separation for context unlabeled data. An important component of this work is the ability to infer the continuous hidden context that contains dynamic properties of the manipulated object, e.g. the mass of the object as illustrated in the experiments.

4 Imitation Learning of Transition Priors

Recall the basic steps we discussed in the introduction: (1) What are the variables? (2) How do these depend upon each other? (3) What are the current goals and constraints? (4) What posterior does this imply on actions?

While Sec. 2 addressed the fourth point of inference, the previous section addressed the second point of learning dependencies between variables from data. Extending this, we have also seen how statistical learning techniques can be used to extract latent or hidden context variables of the problem - which addresses the first point.

In this section, we address a problem which is related to the third point - that of specifying the goals and constraints of the problem. Usually one would consider this to be completely specified externally, e.g., by the engineer. However, often we only have a certain desired behavior vaguely in mind and it becomes

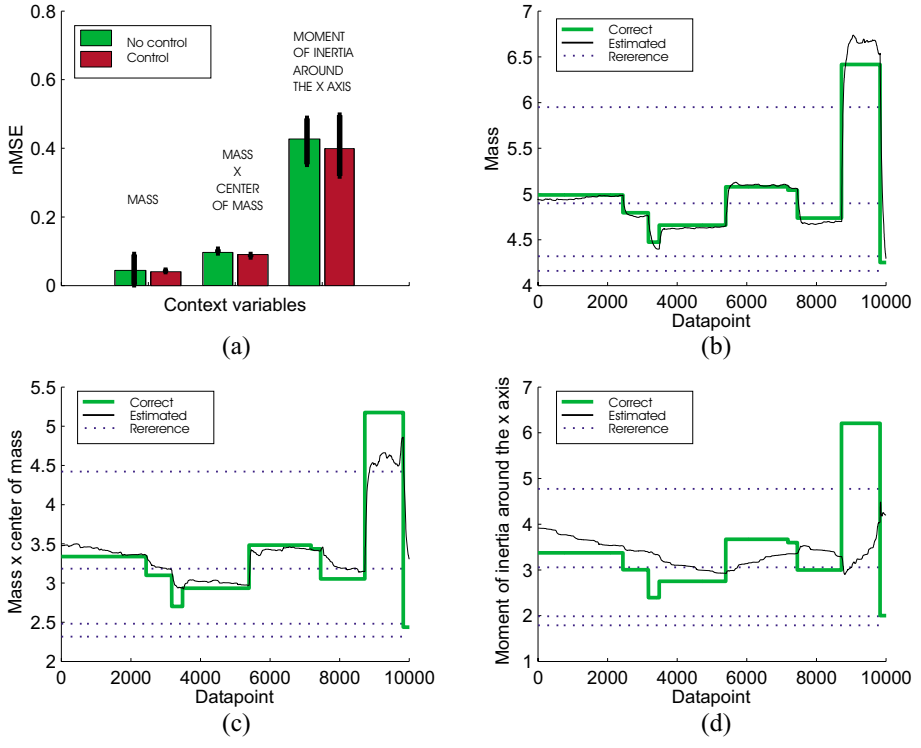


Fig. 13. (a) nMSE of the three contextual variables while using and not using the context estimates for control (b-d) Actual and estimated context variables, along with the values of the context variables of the reference models that were used for deriving the augmented model

rather cumbersome to rigorously define an objective function which will lead to this desired behavior.

Recently there has been rising interest in an alternative to human designed objective functions (alternately, goals and constraints). The idea is to extract, from observed (teacher's) behavior, appropriate qualities that can be utilized as goals or constraint definitions. In our investigations, we focus on what is classically called the nullspace potential. As we have seen in the section on redundant control, a given task constraint does not completely constrain the state trajectory but leaves some unresolved degrees of freedom. This redundancy is most easily resolved by regularization: either by using the W-pseudo inverse in classical control or, in the Bayesian view, by putting a prior $P(\dot{q}) \propto \exp\{-\frac{1}{2}q^T W q\}$ on the joint motion. However, more complex motions require better priors acting on joint motion, e.g., to avoid joint limits. As we have seen, classically this often realized by adding a nullspace movement $h = -\alpha W^{-1}\nabla H(q)$ along the negative gradient of the potential $H(q)$ – which has the Bayesian equivalent in putting a prior $P(\dot{q}) \propto \exp\{-\frac{1}{2}q^T W q + H(q)\}$ on the joint motion (and linearizing $H(q)$

around q). In this section, we discuss learning this potential–aka the transition prior–from observed behavior and hence, being able to extract control policies that generalize over multiple constraints.

4.1 Policies and Potential Functions

A common paradigm for the control of multibody systems such as high DOF manipulators and humanoid robots is to frame the problem as a constrained optimal control problem [16,17,18]. In this paradigm control tasks are formulated as constraints on the system such that some desired behaviour is achieved. In simple systems such as anthropomorphic arms, these constraints often take the form of constraints on the end-effector. For example, the constraints may require that the effector follows a certain trajectory or applies some given force to an object [19].

In a more generic setting, constraints may take a much wider variety of forms. For example in walking or reaching, the constraint may be on the center of mass or tilt of the torso of the walker to prevent over-balancing. Alternately, in contact control problems such as manipulation or grasping, the constraint may require that effectors (such as fingers) maintain a given position on the surface of an object [20]. Also in systems designed to be highly competent and adaptive, such as humanoid robots, behaviour may be subject to a wide variety of constraints [21], usually non-linear in actuator space, and often discontinuous. Consider pouring a cup of water from a bottle; initially constraints will apply to the orientation of the two hands as the water is poured. However once the bottle is empty, the constraint on the orientation of the bottle can be released, representing a discontinuous switch in the constraints on the policy.

The focus of this section is on modelling control policies subject to generic constraints on motion, with the aim of finding policies that can *generalise between different constraints*. In general, learning (unconstrained) policies from constrained motion data is a formidable task. This is due to (i) the non-convexity of observations under different constraints, and; (ii) under any given set of constraints there is *degeneracy* in the set of possible policies that could have produced the constrained movements under observation [22]. However we will show that despite these hard analytical limits, for a certain class of policies, it is possible to find a good approximation of the unconstrained policy given observations under the right conditions. We take advantage of recent work in local dimensionality reduction [23] to show that it is possible to devise a method that (i) given observations under a sufficiently rich set of constraints, reconstructs the fully unconstrained policy; (ii) given observations under an impoverished set of constraints, learns a policy that generalizes well to constraints of a similar class, and; (iii) given 'pathological' constraints will learn a policy that at worst reproduces behaviour subject to the same constraints.

Constrained Policies. Following [24], we consider the learning of autonomous kinematic policies

$$\dot{\mathbf{q}} = \pi(\mathbf{q}(t), \alpha) \tag{44}$$

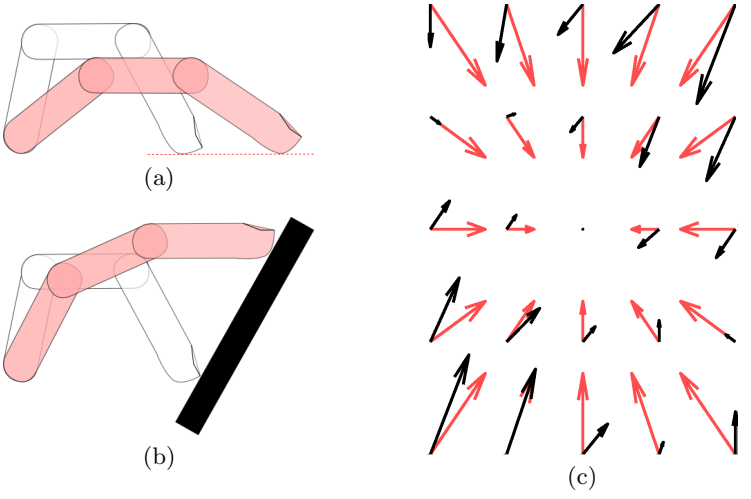


Fig. 14. Illustration of two apparently different behaviours from the same policy: (a) unconstrained movement (b) movement constrained such that the fingertip maintains contact with a surface (black box) (c) the unconstrained (red) and constrained (black) policy over two of the joints of the finger

where $\mathbf{q}$ is some appropriately chosen state-space, $\dot{\mathbf{q}}$ is the desired change in state and α is a vector of parameters determining the behaviour of the policy. The goal of direct policy learning is to approximate the policy (44) as closely as possible [24]. It is usually formulated as a supervised learning problem where it is assumed that we have observations of $\dot{\mathbf{q}}(t)$, $\mathbf{q}(t)$ (often in the form of trajectories), and from these we wish to learn the mapping π. In previous work this has been done by fitting parameterised models in the form of dynamical systems [25,26], non-parametric modelling [18], and probabilistic Bayesian approaches [27,28].

An implicit assumption found in direct policy learning approaches to date is that the data used for training comes from behavioural observations of some *unconstrained* or *consistently constrained* policy. By this it is meant that policy is observed either with no constraints on motion, or where constraints exist, these are static and consistent over observations. For example, consider the learning of a simple policy to extend a jointed finger. In Fig. 14a) the finger is unconstrained and the policy simply moves the joints towards the zero (outstretched) position. On the other hand, in Fig. 14b), an obstacle lies in the path of the finger, so that the finger is constrained to move along the surface of this obstacle. The vector field representation of the two behaviours is shown in Fig. 14c).

In standard approaches to direct policy learning [24,25,26], these two apparently different behaviours would lead to the learning of two separate policies for extending the finger in the two settings. However, the fact that the goals of the two policies are similar ('extend the finger') suggests that in fact the movement stems from the *same policy* under *different constraints*. Viewed like this, instead of learning two separate policies we would rather learn a single policy that generalizes over observations under different constraints.

A *constrained* policy is one for which there are hard restrictions on the movements available to the policy. Mathematically, we say given a set of constraints

$$\mathbf{A}(\mathbf{q}, t)\dot{\mathbf{q}} = \mathbf{0} \tag{45}$$

the policy is projected into the nullspace of those constraints

$$\dot{\mathbf{q}}(t) = \mathbf{N}(\mathbf{q}, t)\pi_{uc}(\mathbf{q}(t)) \tag{46}$$

where $\mathbf{N}(\mathbf{q}, t) \equiv (\mathbf{I} - \mathbf{A}(\mathbf{q}, t)\mathbf{A}(\mathbf{q}, t)^{\dagger})$ is in general a non-linear, time-varying projection operator, $\mathbf{A}(\mathbf{q}, t)$ is some matrix describing the constraint, $\mathbf{A}^{\dagger}$ is the pseudoinverse and $\mathbf{I}$ is the identity matrix. Constrained policies (46) are commonly used for control of redundant degrees of freedom (DOFs) in high-dimensional manipulators [16,17,18], however the formalism is generic and extends to a wide variety of systems, such as team coordination in mobile robots [29].

In this view, the best policy representation of the movements in Fig. 14 is the unconstrained policy π_{uc}, since this is the policy that gives maximal information about the behaviour. Given π_{uc} we can reproduce behaviours such as in Fig. 14 (b) simply by applying the same constraints. Furthermore, if we can find a good approximation of π_{uc} we can even predict behaviour in situations where novel constraints, unseen in the training data, apply.

However, learning the unconstrained policy from observations of constrained movement is a non-trivial task. For example we may not know exactly what constraints were in force at the time of observation. Furthermore there are several analytical restrictions on what information we can hope to recover from constrained motion data [22]. Despite this, we can efficiently uncover the unconstrained policy for the important class of *conservative* policies. In the next section, we characterise these analytical restrictions and show how these can be side-stepped in the case of conservative policies.

Conservative Policies. Learning nullspace policies from constrained motion data is in general a hard problem due to *non-convexity* of observations and *degeneracy* [22].

The non-convexity problem comes from the fact that between observations, or even during the course of a single observation, the constraints may change, resulting in inconsistencies in the training data. For example consider the policy shown in Fig. 14c). In any observation, the observed motion vector $\dot{\mathbf{q}}(t)$ may come from the set of constrained (black) or unconstrained (red) set of vectors. At any given point in the state space we may have multiple observations under different constraints resulting in a set of $\dot{\mathbf{q}}(t)$ at that point. In standard supervised learning algorithms this causes problems since directly training on these observations may result in models that average over the observations. The non-convexity problem then is how to reconcile these multiple conflicting observations to give a consistent policy.

The second problem is degeneracy in the data. This is the problem that for any given set of observations projected into the nullspace of the constraints,

there may be multiple candidate policies that could have produced that movement. This is due to the fact that the projection matrix projects the policy onto a lower dimensional manifold so that motion orthogonal to that manifold is effectively 'zeroed out'. This means that the component of π_{uc} in this direction is undetermined by the observation. In effect the problem is ill-posed in the sense that we are not given sufficient information about the unconstrained policy to guarantee the true policy is learnt.

However, in recent work it was shown [19,22] that for the important special case of *conservative* policies it is possible to use data efficiently to uncover the underlying policy. A conservative policy is a policy that can be described by taking the gradient of a potential function $H(\mathbf{q})$

$$\pi(\mathbf{q}) = -\nabla_{\mathbf{q}} H(\mathbf{q}). \tag{47}$$

Conservative policies can be thought of as policies that greedily optimise the potential function at every time step [30]. Such policies encode attractor landscapes where their minima correspond to stable attractors; in the finger example, the $\mathbf{q} = \mathbf{0}$ point would correspond to such a minimum. Conservative policies are commonly used in control of redundant DOFs in manipulators [16,17,18].

4.2 Learning Nullspace Policies through Local Model Alignment

If the policy under observation is conservative, an elegant solution to solving the non-convexity and degeneracy problems is to model the policy through its *potential function* [19,22] rather than modelling it directly. The advantage of this is twofold. Firstly, due to the local linearity of the projection operator $\mathbf{N}(\mathbf{q}, t)$ the conservative policy (47) remains *locally conservative* in the lower dimensional nullspace. We can use numerical line integration to estimate the form of the potential along the trajectories [19,22]. Secondly, the potential function is a scalar function and thus gives a compact representation of the policy. Crucially, this means that the problem of reconciling conflicting n-dimensional vector observations is reduced to finding a function $\hat{H}(\mathbf{q})$ where the (1-dimensional) prediction is consistent at any given point $\mathbf{q}$. Next, we propose a method for modelling the potential on a trajectory-wise basis and for consolidating models from multiple trajectories.

Estimating the Potential along Single Trajectories. A method to model the potential along trajectory is to use an integration scheme such as the Euler integration, which involves the first order approximation

$$H(\mathbf{q}_{t+1}) \approx H(\mathbf{q}_t) + (\mathbf{q}_{t+1} - \mathbf{q}_t)^T \mathbf{N}(\mathbf{q}_t)\pi(\mathbf{q}_t) \tag{48}$$

Please note that for steps $\mathbf{q}_t \rightarrow \mathbf{q}_{t+1}$ that follow the projected policy, $(\mathbf{q}_{t+1} - \mathbf{q}_t) = \mathbf{N}(\mathbf{q}_t)\pi(\mathbf{q})$, so we can actually write

$$H(\mathbf{q}_{t+1}) \approx H(\mathbf{q}_t) - |\, \mathbf{q}_{t+1} - \mathbf{q}_t \,|^2. \tag{49}$$

We use this approximation to generate estimates $\hat{H}(\mathbf{q}_i)$ of the potential along any given trajectory $\mathbf{q}_1, \mathbf{q}_2 \dots \mathbf{q}_N$ in the following way: We set $\hat{H}_1 = \hat{H}(\mathbf{q}_1)$ to an arbitrary value and then iteratively assign $\hat{H}_{i+1} := \hat{H}_i - |\,\mathbf{q}_{i+1} - \mathbf{q}_i\,|^2$ for the remaining points in the trajectory.

Note that an arbitrary constant can be added to the potential function without changing the policy. Therefore, 'local' potentials that we estimate along different trajectories need to be *aligned* in a way that their function value matches in intersecting regions. We'll turn to this problem next.

Constructing the Global Potential Function. Let us assume we are given K trajectories $\mathbf{Q}_k = (\mathbf{q}_{k1}, \mathbf{q}_{k2} \dots \mathbf{q}_{kN_k})$ and corresponding point-wise estimates $\hat{\mathbf{H}}_k = (\hat{H}_{k1}, \hat{H}_{k2} \dots \hat{H}_{kN_k})$ of the potential, as provided from the Euler integration just described. In a first step, we fit a function model $f_k(\mathbf{q})$ of the potential to each tuple $(\mathbf{Q}_k, \hat{\mathbf{H}}_k)$, such that $f_k(\mathbf{q}_i) \approx \hat{H}_{ki}$. To keep things simple, we choose a nearest-neighbour regression model, i.e.,

$$f_k(\mathbf{q}) = H_{ki^*} \quad , \quad i^* = \arg\min_i |\,\mathbf{q} - \mathbf{q}_{ki}\,|^2. \tag{50}$$

Since we wish to combine the models to a global potential function, we need to define some function for weighting the outputs of the different models. For the nearest-neighbour algorithm, we choose to use a Gaussian kernel

$$w_k(\mathbf{q}) = \exp\left[-\frac{1}{2\sigma^2} \min_i |\,\mathbf{q} - \mathbf{q}_{ki}\,|^2\right]. \tag{51}$$

From these weights we can calculate responsibilities

$$r_k(\mathbf{q}) = \frac{w_k(\mathbf{q})}{\sum_{i=1}^K w_i(\mathbf{q})} \tag{52}$$

and a (naive) global prediction $f(\mathbf{q}) = \sum_{k=1}^K r_k(\mathbf{q}) f_k(\mathbf{q})$ of the potential at $\mathbf{q}$.

However, as already stated, the potential is only defined up to an additive constant, and most importantly this constant can vary from one local model to another. This means that we first have to shift the models by adding some *offset* to their estimates of the potential, such that all local models are *in good agreement* about the global potential at any number of states $\mathbf{q}$.

We follow the methodology of non-linear dimensionality reduction [23] as used to align multiple local PCA models into a common low-dimensional space. In analogy to the PCA-alignment method [23], we augment our local potential models $f_k(\cdot)$ by a scalar offset b_k and solve for the corresponding objective function:

$$E(b_1 \dots b_K) = \frac{1}{2} \sum_{m=1}^M \sum_{k=1}^K \sum_{j=1}^K r_k(\mathbf{q}_m) r_j(\mathbf{q}_m) \left((f_k(\mathbf{q}_m) + b_k) - (f_j(\mathbf{q}_m) + b_j)\right)^2, \tag{53}$$

or, in a slightly shorter form,

$$E(\mathbf{b}) = \frac{1}{2} \sum_{m,k,j} r_{km} r_{jm} \left(f_{km} + b_k - f_{jm} - b_j\right)^2. \tag{54}$$

Here, $\sum_m$ denotes a summation over the complete dataset, that is, over all points from all trajectories ($M = \sum_{k=1}^{K} N_k$). Solving the above objective function for the optimal shift $\mathbf{b}_{opt}$ yields the alignment necessary for global learning. For details of the solution for detecting the optimal alignment offset, readers are referred to [31].

Since we restrict ourselves to using simple nearest neighbor (NN) regression for the local potential models in this paper, the only open parameter of our algorithm is σ^2, i.e., the kernel parameter used for calculating the responsibilities (51). A too large choice of this parameter will over-smooth the potential, because the NN regression model basically predicts a locally constant potential, but at the same time trajectories will have relatively high responsibilities for even far apart points $\mathbf{x}$ in state space. On the other hand, a too small value of σ^2 might lead to *weakly connected trajectories*: If a particular trajectory does not make any close approach to other trajectories in the set, the quick drop-off of its responsibility implies that it will not contribute to the alignment error (based on pairs of significant responsibility), which in turn implies that its own alignment – the value of its offset – does not matter much. We again refer the reader to [31] for details of the detecting and eliminating such outlier trajectories.

Learning the Global Model. After calculating optimal offsets $\mathbf{b}_{opt}$ and cleaning the dataset from outliers, we can learn a global model $f(\mathbf{q})$ of the potential using any regression algorithm. Here, we choose Locally Weighted Projection Regression (LWPR) [9] because it has been demonstrated to perform well in cases where the data lies on low-dimensional manifolds in a high-dimensional space, which matches our problem of learning the potential from a set of trajectories. As the training data for LWPR, we use all non-outlier trajectories and their estimated potentials as given by the Euler integration *plus* their optimal offset, that is, the input-output tuples

$$\left\{ (\mathbf{q}_{kn}, \hat{H}_{kn} + b_k^{opt}) \mid k \in \mathcal{K}, n \in \{1 \dots N_k\} \right\}, \tag{55}$$

where $\mathcal{K}$ denotes the set of indices of non-outlier trajectories. Once we have learned the model $f(\mathbf{q})$ of the potential, we can take derivatives to estimate the unconstrained policy $\hat{\pi}(\mathbf{q}) = -\nabla_{\mathbf{q}} f(\mathbf{q})$ or use the potential function directly as described in the beginning of Sec. 4.

4.3 Experiments in Direct Policy Learning

To explore the performance of the algorithm, we perform experiments on data from autonomous kinematic control policies [24] applied to three simulated plants, including a physically realistic simulation of the 27 DOF humanoid robot ASIMO [21]. However, to illustrate the key concepts involved, we first discuss results from two simplified problems[2] controlled according to the same generic framework.

[2] In fact even these 'simplified' problems are relevant to constrained policy learning in low dimensional task space representations, for example in end-effector space of an arm.

Selection of Smoothing Parameter. For simplicity, in all our experiments we used the same heuristics for selecting the smoothing parameter σ^2 to match the scale of typical distances in the datasets. In particular, we first calculated the distances between any two trajectories $k, j \in \{1 \dots K\}$ in the set as the distances between their closest points

$$d_{kj} = \min \left\{ \mid \mathbf{q}_{kn} - \mathbf{q}_{jm} \mid^2 \mid n, m \in \{1 \dots N\} \right\}, \tag{56}$$

and also the distances to the closest trajectory

$$d_k^{min} = \min \left\{ d_{kj} \mid j \neq k \right\}. \tag{57}$$

We then consider three choices for σ^2, which we refer to as 'narrow', 'wide' and 'medium':

$$\sigma_{nar}^2 = \text{median} \left\{ d_k^{min} \mid k \in \{1 \dots K\} \right\} \tag{58}$$

$$\sigma_{wid}^2 = \text{median} \left\{ d_{jk} \mid j, k \in \{1 \dots K\}, j \neq k \right\} \tag{59}$$

$$\sigma_{med}^2 = \sqrt{\sigma_{nar}^2 \sigma_{wid}^2}. \tag{60}$$

Toy Example. The toy example consists of a two-dimensional system with a quadratic nullspace potential subject to discontinuously switching task constraints. Specifically, the potential function is given by

$$H(\mathbf{q}) = \mathbf{q}^T \mathbf{W} \mathbf{q} \tag{61}$$

where $\mathbf{W}$ is some square weighting matrix which we set to $0.05\mathbf{I}$. Data was collected by recording trajectories generated by the policy from a start state distribution Q_0. During the trajectories, the policy was subjected to random 1-D constraints:

$$\mathbf{A}(\mathbf{q}, t) = (\alpha_1, \alpha_2) \equiv \alpha \tag{62}$$

where the $\alpha_{1,2}$ were drawn from a normal distribution, $\alpha_i = N(0, 1)$. The constraints mean that motion is constrained in the direction orthogonal to the vector α in state space. To increase the complexity of the problem, the constraints were randomly switched during trajectories by re-sampling α twice at regular intervals during the trajectory. This switches the direction in which motion is constrained as can be seen by the sharp turns in the trajectories.

Figure 15 shows an example of our algorithm at work for a set of $K = 40$ trajectories of length $N = 40$ for the toy system. The raw data as a set of trajectories through the two-dimensional state space is shown in panel (a), whereas panel (b) additionally depicts the local potential models as estimated from the Euler integration prior to alignment. Each local model has an arbitrary offset against the true potential so there are inconsistencies between the predictions from each local model. Figure 15(c) shows the trajectories after alignment, already revealing the structure of the parabola. At this point, the outlier detection scheme has identified three trajectories as being weakly connected to the

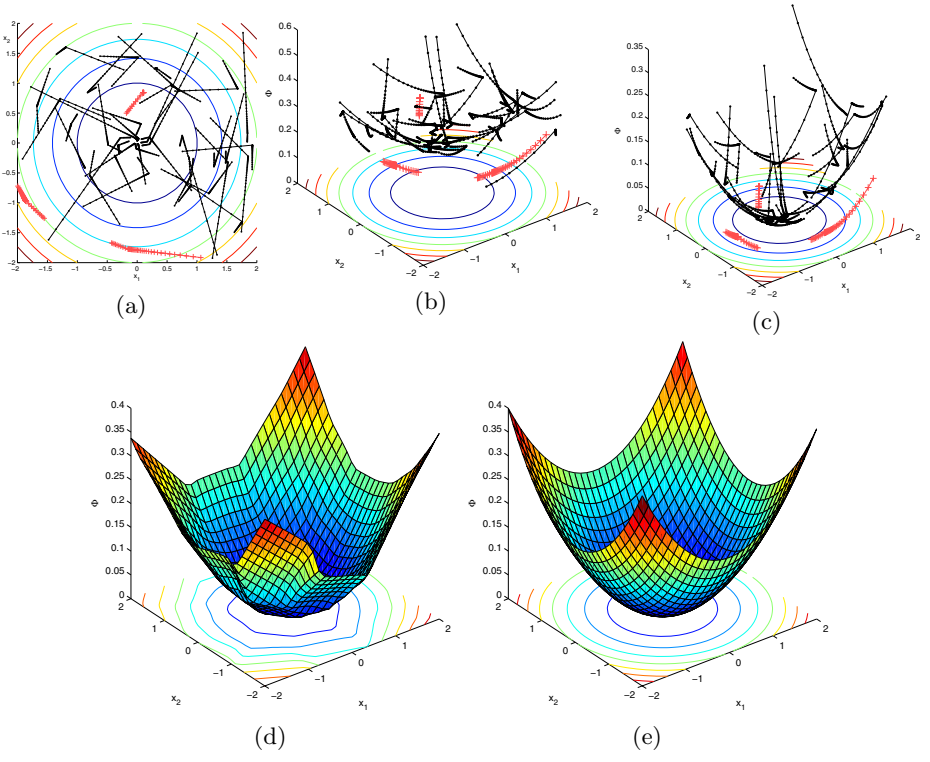


Fig. 15. Top: a) Toy data trajectories (2-D) and contour of true potential. Estimated potential along the trajectories before (b) and after (c) alignment. Trajectories detected as difficult to align 'outliers' are shown by light crosses. Bottom: Learnt (d) and true (e) potential function after training on the aligned trajectories.

remaining set. In Fig. 15(a), we can see that the outliers are indeed the only trajectories that do not have any intersection with neighboring trajectories. At the 'narrow' length scale determined by the smoothing parameter (58), they are hard to align properly, and need to be discarded before learning the global model. Finally, Fig.15(d) shows the global model $f(\mathbf{q})$ of the potential that was trained on the aligned trajectories, which is clearly a good approximation of the true parabolic potential shown in Fig.15(e). For a more thorough evaluation, we repeated this experiment on 100 datasets and evaluated

- the nMSE of the aligned potential, which measures the difference between $\hat{H}_{kn} + b_k$ and the true potential H,
- the nMSE of the learnt potential, measuring the difference between $f(\cdot)$ and $H(\cdot)$,
- the normalised unconstrained policy error (UPE), depending on the difference between $\hat{\pi} = \nabla f$ and $\pi = \nabla H$,

Table 3. Error and outlier statistics (nMSE over 100 data sets) for the experiment on 2-D toy data

Setup	σ^2	Alignment (nMSE)	Potential (nMSE)	UPE (nMSE)	CPE (nMSE)	Outliers discarded (%)
Parabola	narrow	0.0039	0.0044	0.0452	0.0211	19.15
K=40	medium	0.0178	0.0168	0.0798	0.0186	0.25
N=40	wide	0.3494	0.3091	0.5177	0.0930	0
Sinusoidal	narrow	0.0012	0.0022	0.1132	0.0482	52.67
K=40	medium	0.0634	0.0623	0.1359	0.0343	0.77
N=40	wide	0.6231	0.5653	0.8397	0.2538	0
Sinusoidal	narrow	0.0006	0.0014	0.0550	0.0270	27.80
K=100	medium	0.0100	0.0097	0.0678	0.0255	0.15
N=100	wide	0.6228	0.5367	0.6972	0.2304	0

- the normalised constrained policy error (CPE), which is the difference between $\mathbf{N}\hat{\pi}$ and $\mathbf{N}\pi$, and finally
- the percentage of trajectories discarded as outliers.

We did so for our three different choices of σ^2 given in (58-60). We also repeated the experiment using a sinusoidal potential function

$$H_s(\mathbf{q}) = 0.1 \sin(q_1) \cos(q_2) \tag{63}$$

with the same amount of data, and using $K = 100$ trajectories of length $N = 100$ for each dataset.

Table 3 summarises the results. Firstly, we can see that the 'wide' choice for σ^2 leads to large error values which are due to over-smoothing. Using the narrow σ^2, we retrieve very small errors at the cost of discarding quite a lot of trajectories[3], and the medium choice seems to strike a reasonable balance especially with respect to the UPE and CPE statistics.

Secondly, when comparing the results for the parabolic and sinusoidal potentials, we can see that the latter, more complex potential (with multiple sinks) requires much more data. With only 40 trajectories and 40 points each, most of the datasets are too disrupted to learn a reasonable potential model. While at the narrow length scale (4th row), on average more than half of the dataset is discarded, even the medium length scale (5th row) over-smooths the subtleties of the underlying potential.

Finally, the constrained policy error (CPE) is always much lower than the UPE, which follows naturally when training on data containing those very movement constraints. Still, with a reasonable amount of data, even the unconstrained policy can be modelled with remarkable accuracy.

[3] Please note that we also discard the outliers for evaluating the error statistics – we can hardly expect to observe good performance in regions where the learnt model $f(\mathbf{q})$ has seen no data.

Table 4. Error and outlier statistics for the three link arm (TLA) and whole body motion (WBM) controller

Plant, σ^2	Alignment (nMSE)	Potential (nMSE)	UPE (nMSE)	CPE (nMSE)	Outliers discarded (%)
TLA, narrow	0.2149	0.2104	0.3908	0.0526	18.07
TLA, medium	0.6029	0.6012	0.6526	0.0386	0
WBM, narrow	0.0007	0.0007	0.0896	0.0816	31.15
WBM, medium	0.0016	0.0016	0.1424	0.0778	0

Three Link Arm. The two goals of our second set of experiments were (i) to characterise how well the algorithm scaled to more complex, realistic constraints and (ii) to characterise how well the learnt policies generalised over different constraints. For this we used a planar three-link arm (TLA) with revolute joints and unit link lengths. Using the TLA allowed us to set up a much richer variety of constraints, such as constraints on kinematic variables for example the end-effector (hand) position and orientation. Here we report results for a set of intuitively appealing constraints, that is a set of planar constraints on the hand. This kind of constraint occurs in contact-based behaviour [20], for example in writing, where the hand must maintain contact with a planar surface[4] such as a table top.

Data was collected by recording $K = 100$ trajectories of length $N = 100$ from a random distribution of start states. For ease of comparison with the 2-D system, the nullspace policy was chosen to optimise the same quadratic potential (61). The policy was constrained through the matrix

$$\mathbf{A}(\mathbf{q}, t) = \hat{\mathbf{n}}^T \mathbf{J}_{hand}(\mathbf{q}) \tag{64}$$

where $\hat{\mathbf{n}}$ is a unit vector normal to the hand-space plane and $\mathbf{J}_{hand}(\mathbf{q})$ is the hand Jacobian. The constraints (64) are highly nonlinear in the joint space where the policy is operating. Finally, to simulate observations under different constraints, the orientation of the hand-space plane was changed for each trajectory by drawing $\hat{\mathbf{n}}$ from a uniform random distribution of two-dimensional unit vectors $D_{\hat{\mathbf{n}}}$. We then used our algorithm to learn the nullspace potential. The results of learning are shown in Table 4.

Generalising over Unseen Constraints

Our first test was to look at the performance of the algorithm in finding a policy that generalises over *unseen* constraints. To do this we defined two new 'test' constraints and evaluated the CPE using these constraints in place of the training data constraints. The test constraints chosen were (i) an unseen planar constraint on the hand (i.e. we set $\hat{\mathbf{n}} = \hat{\mathbf{n}}_{test}$ where $\hat{\mathbf{n}}_{test}$ is drawn from $D_{\hat{\mathbf{n}}}$, and; (ii) constraining the hand *orientation* during motion. This latter constraint is

[4] Such constraints also need not be linear, and can be generalised to any shaped surface.

Table 5. Constrained policy nMSE for unseen constraints on the three-link arm. Values are mean±s.d. over 100 data sets.

Constraint	CPE
Training	0.0526 ± 0.0192
Unseen hand-space plane	0.0736 ± 0.0492
Hand Orientation	0.1053 ± 0.0524
Unconstrained	0.3908 ± 0.2277

qualitatively similar to the training data constraints (i.e., a 1-D constraint on the hand) but produces visibly different behaviour, in terms of hand- and joint-space trajectories.

Table 5 gives a comparison of the normalised policy error evaluated on the unconstrained policy, the constrained policy, and the policy subject to the two test constraints over 100 data sets. The first thing to note is that the algorithm shows good performance for the CPE evaluated on the training data constraints, indicating a minimum guarantee on performance, namely that the learnt potential will at worst be consistent with the training data. Secondly, the progression of error over the two unseen constraints coincides with the extent to which they are similar to the constraints in the training data. This confirms our intuition that though we can generalise over different constraints, this becomes increasingly difficult as these depart from those observed. Finally, the unconstrained policy error indicates that for some reason the algorithm is having problems finding the fully unconstrained policy in this case. We investigate this issue more closely in the next section.

Unconstrained Policy Error. The reason for the poor performance of the algorithm in predicting the unconstrained policy (ref. Table 5) becomes clear if we analyse the effect of the constraints on the movement of the arm. Fig 16(a) shows the training data trajectories through the three joints of the arm. It is clear that owing to the constraints on the arm, the policy no longer reaches the point attractor at $\mathbf{q} = \mathbf{0}$, but instead reaches a line in joint space (shown in black). This 'line attractor' represents the minimum of the potential that can be reached without breaking the constraints. Furthermore, it seems that away from this line, there are few points where trajectories come close to one another or intersect. This means that the algorithm gets little or no information about how the potential changes in this direction.

This is confirmed by comparing how the UPE and CPE changes as we move along the attractor, and radially outward from it. To demonstrate this, we evaluated the potential nMSE, UPE and CPE on data contained within different regions of the state space. Firstly, we looked at how the error changed on data points contained between two planes normal to the line at distance d from the point attractor $\mathbf{q} = \mathbf{0}$ (Fig 16(b), dashed lines), and plotted it with increasing d (Fig 16(d)). We can see that close to $\mathbf{q} = \mathbf{0}$, the potential nMSE and UPE start low but increase rapidly for large d. On the other hand the CPE stays relatively constant over the entire set.

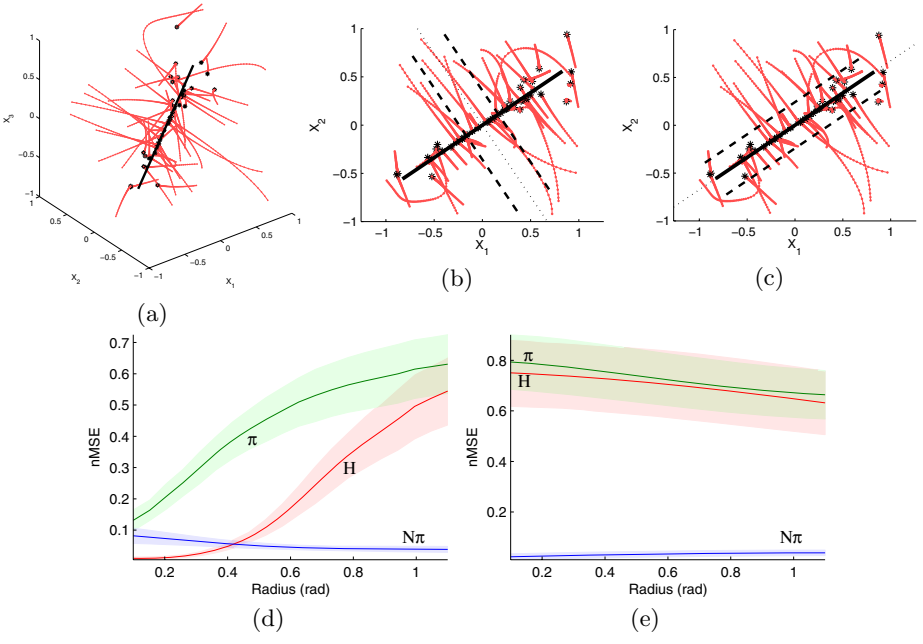


Fig. 16. (a) Trajectories in state-space for the TLA subject to random planar constraints on the hand. (b) and (c) show projections onto the first two joints of the arm, and also indicate the line attractor (solid black line). We sampled the nMSE at increasing distances along the line (b) and radially outward from it (c). Plots (d) and (e) depict the cumulative nMSE of the potential H, policy π, and constrained policy ($\mathbf{N}\pi$) as a function of the distance measures from (b) and (c), respectively.

Secondly, we looked at how the errors change as we move radially outward. For this, we evaluated errors on data contained within a cylinder of radius l centred on the line attractor (Fig 16(c), dashed lines). Fig 16(e) shows the change in error with increasing radius l. Again the CPE remains constant. This time, however, the potential nMSE and UPE are high even at small l. This indicates that the points at the two ends of the line are contributing most of the error.

We can therefore say that the seemingly poor performance of our algorithm on this problem is due to the adverse constraints in the training data. The constraints do not permit motion in the direction of the line attractor, so we cannot hope to recover the potential function along that direction. However, the good generalisation of the learnt policy over unseen constraints indicates that the algorithm is performing reasonably well despite these adverse conditions.

ASIMO Data. Using a realistic simulation [21] of the humanoid robot ASIMO (refer Fig. 17), we tested the scalability of our approach for learning in high dimensions. We collected data from the nullspace policy subject to a mix of constraints, including random planar constraints (in hand-space) on the two

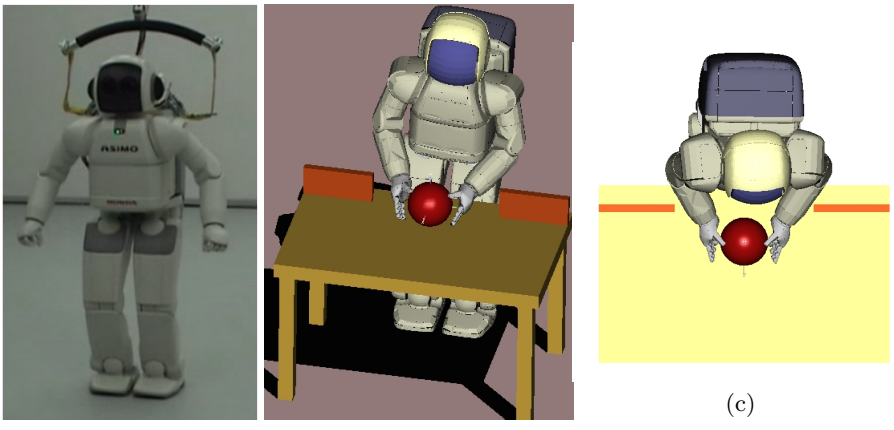


Fig. 17. (a) The Humanoid ASIMO, (b) front view and (c) top view of a realistic VRML simulation of the robot with full kinematics and dynamics

hands of the robot as in (64), and constraints that fixed the position of the hands in hand-space. The latter occurs in a variety of behaviours. For example in co-operative or bi-manual manipulation tasks, one of the hands may be constrained to hold the manipulated object in position, while the nullspace policy acts to move the rest of the system into a comfortable posture [21].

Table 4 shows the learning performance of the algorithm subject to these constraints. The potential and the unconstrained policy errors are remarkably good and even out-perform those of the lower dimensional systems. We attribute this remarkable performance to the constraints on motion being much lower dimensional than the 27 DOFs of the policy. This means that there is a high chance that many of the trajectories reach the point attractor of the policy, which simplifies the alignment and the learning of the potential.

4.4 Conclusion

In this section, we demonstrated a novel approach to direct learning of conservative policies from constrained motion data. The method is fast and data-efficient, and scales to complex constraints in high-dimensional movement systems. The core ingredient is an algorithm for aligning local models of the potential, which leads to a convex optimisation problem. Ultimately, the ability to learn the nullspace potential depends on the constraints. Given a pathological set of constraints, one can never hope to recover the potential. However, we suggest a paradigm whereby motion data under different constraints can be combined to learn a potential that is consistent with the observations. With a reasonably rich set of constraints, one can recover the nullspace potential with high accuracy, and then, use this to generalise and predict behaviour under different constraints.

Acknowledgements. This work was supported by the Microsoft/Royal Academy of Engineering Senior Research Fellowship to SV, the EU FP6 SENSOPAC grant to SV, the EPSRC HONDA CASE studentship to MH, the Greek State PhD scholarship to GP and the DFG Emmy Noether grant to MT.

References

1. Peters, J., Mistry, M., Udwadia, F.E., Cory, R., Nakanishi, J., Schaal, S.: A unifying framework for the control of robotics systems. In: IEEE Int. Conf. on Intelligent Robots and Systems (IROS 2005), pp. 1824–1831 (2005)
2. Nakamura, Y., Hanafusa, H.: Inverse kinematic solutions with singularity robustness for robot manipulator control. Journal of Dynamic Systems, Measurement and Control 108 (1986)
3. Baerlocher, P., Boulic, R.: An inverse kinematic architecture enforcing an arbitrary number of strict priority levels. The Visual Computer (2004)
4. Todorov, E.: Optimal control theory. In: Doya, K. (ed.) Bayesian Brain: Probabilistic Approaches to Neural Coding, pp. 269–298. MIT Press, Cambridge (2006)
5. Platt, R., Fagg, A., Grupen, R.: Nullspace composition of control laws for grasping. In: Proceedings of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems, Lausanne, Switzerland (2002)
6. Ijspeert, A.J., Nakanishi, J., Schaal, S.: Learning attractor landscapes for learning motor primitives. In: Advances in Neural Information Processing Systems, vol. 15, pp. 1523–1530. MIT Press, Cambridge (2003)
7. Schaal, S., Peters, J., Nakanishi, J., Ijspeert, A.: Control, planning, learning, and imitation with dynamic movement primitives. In: Workshop on Bilateral Paradigms on Humans and Humanoids, IEEE Int. Conf. on Intelligent Robots and Systems, Las Vegas, NV (2003)
8. Nakanishi, J., Morimoto, J., Endo, G., Cheng, G., S., Schaal, K.M.: Learning from demonstration and adaptation of biped locomotion with dynamical movement primitives. In: Workshop on Robot Learning by Demonstration, IEEE Int. Conf. on Intelligent Robots and Systems (2003)
9. Vijayakumar, S., D'Souza, A., Schaal, S.: Incremental online learning in high dimensions. Neural Computation 17, 2602–2634 (2005)
10. Klanke, S., Vijayakumar, S., Schaal, S.: A library for locally weighted projection regression. Journal of Machine Learning Research (2008)
11. Roweis, S., Ghahramani, Z.: 6. In: Haykin, S. (ed.) Learning Nonlinear Dynamical Systems using the EM Algorithm, pp. 175–220. Wiley, Chichester (2001)
12. Briegel, T., Tresp, V.: Fisher scoring and a mixture of modes approach for approximate inference and learning in nonlinear state space models (1999)
13. de Freitas, J., Niranjan, M., Gee, A.: Nonlinear state space estimation with neural networks and the em algorithm. Technical report (1999)
14. Sciavicco, L., Siciliano, B.: Modelling and Control of Robot Manipulators. Springer, Heidelberg (2000)
15. Craig, J.J.: Introduction to Robotics: Mechanics and Control. Pearson Prentice Hall, London (2005)
16. Liégeois, A.: Automatic supervisory control of the configuration and behavior of multibody mechanisms. IEEE Trans. Systems, Man, and Cybernetics SMC-7, 245–250 (1977)

17. Khatib, O.: A unified approach for motion and force control of robot manipulators: The operational space formulation. IEEE Journal of Robotics and Automation RA-3(1), 43–53 (1987)
18. Peters, J., Mistry, M., Udwadia, F.E., Nakanishi, J., Schaal, S.: A unifying framework for robot control with redundant DOFs. Autonomous Robots Journal 24, 1–12 (2008)
19. Howard, M., Gienger, M., Goerick, C., Vijayakumar, S.: Learning utility surfaces for movement selection. In: IEEE International Conference on Robotics and Biomimetics (ROBIO) (2006)
20. Park, J., Khatib, O.: Contact consistent control framework for humanoid robots. In: Proc. IEEE Int. Conf. on Robotics and Automation (ICRA) (May 2006)
21. Gienger, M., Janssen, H., Goerick, C.: Task-oriented whole body motion for humanoid robots. In: 5th IEEE-RAS International Conference on Humanoid Robots, 2005, December 5, 2005, pp. 238–244 (2005)
22. Howard, M., Vijayakumar, S.: Reconstructing null-space policies subject to dynamic task constraints in redundant manipulators. In: Workshop on Robotics and Mathematics (RoboMat) (September 2007)
23. Verbeek, J.J., Roweis, S.T., Vlassis, N.: Non-linear CCA and PCA by alignment of local models. In: Advances in Neural Information Processing Systems, vol. 16. MIT Press, Cambridge (2004)
24. Schaal, S., Ijspeert, A., Billard, A.: Computational approaches to motor learning by imitation. In: The Neuroscience of Social Interaction, pp. 199–218. Oxford University Press, Oxford (2004)
25. Ijspeert, A.J., Nakanishi, J., Schaal, S.: Learning attractor landscapes for learning motor primitives. In: Becker, S., Thrun, S., Obermayer, K. (eds.) Advances in Neural Information Processing Systems, vol. 15, pp. 1523–1530. MIT Press, Cambridge (2003)
26. Ijspeert, A.J., Nakanishi, J., Schaal, S.: Movement imitation with nonlinear dynamical systems in humanoid robots. In: Proc. IEEE International Conference on Robotics and Automation (ICRA), pp. 1398–1403 (2002)
27. Grimes, D.B., Chalodhorn, R., Rao, R.P.N.: Dynamic imitation in a humanoid robot through nonparametric probabilistic inference. In: Proceedings of Robotics: Science and Systems (RSS 2006). MIT Press, Cambridge (2006)
28. Grimes, D.B., Rashid, D.R., Rao, R.P.N.: Learning nonparametric models for probabilistic imitation. In: Advances in Neural Information Processing Systems (NIPS 2006), vol. 19. MIT Press, Cambridge (2007)
29. Antonelli, G., Arrichiello, F., Chiaverini, S.: The null-space-based behavioral control for soccer-playing mobile robots. Proceedings, pp. 1257–1262 (2005)
30. Nakamura, Y.: Advanced Robotics: Redundancy and Optimization. Addison Wesley, Reading (1991)
31. Howard, M., Klanke, S., VIjayakumar, S.: Learning nullspace potentials from constrained motion. In: Proc. IEEE International Conference on Intelligent Robots and Systems (IROS) (2008)