FP7-600716

Whole-Body Compliant Dynamical Contacts in Cognitive Humanoids

D4.2
Learning of tasks with multiple contacts by imitation and reinforcement learning

Editor(s)	Elmar Rueckert[1] and Jan Peters[1,2]
Responsible Partner	TUD
Affiliations	[1] Intelligent Autonomous Systems Lab, Technische Universität Darmstadt, 64289 Darmstadt, Germany. [2] Robot Learning Group, Max-Planck Institute for Intelligent Systems, Tuebingen, Germany.
Status-Version:	Draft-1.0
Date:	Feb. 28, 2015
EC Distribution:	Consortium
Project Number:	600716
Project Title:	Whole-Body Compliant Dynamical Contacts in Cognitive Humanoids

Title of Deliverable:	Learning of tasks with multiple contacts by imitation and reinforcement learning
Date of delivery to the EC:	28/2/2015

Workpackage responsible for the Deliverable	WP4
Editor(s):	Jan Peters and Elmar Rueckert
Contributor(s):	Elmar Rueckert, Alex Paraschos, Roberto Calandra, Oliver Kroemer, Serena Invaldi, Jan Peters (TUD) / Jernej Camernik, Jan Babic (JSI)
Reviewer(s):	
Approved by:	All Partners

Abstract	The scope of the current deliverable is to present the results on generalizing and improving elementary tasks with contacts.
Keyword List:	contacts, inverse dynamics model learning, probabilistic movement representations, reinforcement learning

Document Revision History

Version	Date	Description	Author
v. 0.1	Feb. 04	Initial Draft	Elmar Rueckert
v. 1.0	Feb. 28	Final	Elmar Rueckert

Table of Contents

Chapter 1

Introduction

This deliverable presents results of task T4.3 at the end of the second year. The achieved results are briefly discussed with respect to the task description from the Technical Annex.

T4.3 Generalizing and Improving Elementary Tasks with Contacts. The architecture from WP3 requires a variety of tasks in order to be useful in a large number of different scenarios. T4.3 aims at filling this void by automatically generating new tasks from data. Using both models and data traces from WP2, elementary tasks are acquired by imitation learning and transferred to novel situations using dynamic systems. In this case, the well-known formulation by (Ijspeert et al., 2002) will need to be reformulated to be capable of taking contact explicitly into account. Using the recently developed relative entropy inverse reinforcement learning techniques (Boularias et al., 2011), these tasks can be acquired quickly while at the same time yielding an explanation of the tasks functionality in form of an estimate of the teacher's cost function. Subsequently, the robot system will self-improve on its' current hardware in order to achieve mastery at the task. The procedural steps of this work package are the following:

- Data sets from human behavioral experiments are used to generate elementary tasks such as reaching for objects, pushing, pulling, and standing up.

  In collaboration with JSI, TUD investigated weather supporting contacts in human arm reaching tasks are planed or an effect of a reactive controller. We found that contacts are task related and thus planed. Further, trunk movements are less affected by feedback compared to the lighter arms. Based on these findings TUD will develop combined feedforward and feedback control laws and test them on the iCub robot. A paper is currently in progress of writing.

- These tasks are used to instantiate an elementary task library by imitation learning, yielding both functional elementary tasks as well as an associated cost function.

  In movement libraries, a controller needs to activate suitable primitives based on contextual cues or external events. TUD has extended their work on probabilistic movement primitives to simultaneously learn a library of movement primitives and control variables to activate the primitives. This work was published at an international robotics conference [69] and is discussed in Chapter 3.

As a first step towards understanding the stability of supporting contacts in humanoid robot control, TUD developed a probabilistic kernel based model to predict the stability of object configurations. This work was published at an international robotics conference [45] and is presented in Chapter 4.

- The tasks are refined by reinforcement leaning such that the task will work also on the iCub despite differences in kinematics and mechanics.

- The task parameters of the local task are passed on to the controller from WP3.

  Controller need to be able to compensate for external contact forces to either decelerate to avoid collisions or to push objects in a controller manner, e.g., through tactile feedback controller. TUD investigated how the tactile skin of the iCub can be used to learn inverse dynamics models in the presence of contacts. This work was published an international robotics conference [8] and a pre-print is presented in Chapter 5. In the subsequent chapter it is shown how learned inverse dynamics models can be used to improve torque control of the iCub. This work is currently in progress of writing.

Chapter 2

Executive Summary

In this deliverable, four studies relevant for improving and generalizing elementary tasks with contacts are presented. The first work demonstrates how multiple movement primitives as well as their activations can be learned from human demonstrations. In a second study, TUD demonstrates how the stability of object configurations can be modeled and used for predictions. This work will be used in future to search for stable supporting contact configurations in humanoid robot motions. In the last two chapters, mixtures of experts with Gaussian processes are used to learn inverse dynamics models from tactile skins in the iCub and how to use these models for improving the performance of torque controller.

Chapter 3

Extracting Low-Dimensional Control Variables for Movement Primitives (TUD)

Abstract

Movement primitives (MPs) provide a powerful framework for data driven movement generation that has been successfully applied for learning from demonstrations and robot reinforcement learning. In robotics we often want to solve a multitude of different, but related tasks. As the parameters of the primitives are typically high dimensional, a common practice for the generalization of movement primitives to new tasks is to adapt only a small set of control variables, also called meta parameters, of the primitive. Yet, for most MP representations, the encoding of these control variables is pre-coded in the representation and can not be adapted to the considered tasks. In this paper, we want to learn the encoding of task-specific control variables also from data instead of relying on fixed meta-parameter representations. We use hierarchical Bayesian models (HBMs) to estimate a low dimensional latent variable model for probabilistic movement primitives (ProMPs), which is a recent movement primitive representation. We show on two real robot datasets that ProMPs based on HBMs outperform standard ProMPs in terms of generalization and learning from a small amount of data and also allows for an intuitive analysis of the movement. We also extend our HBM by a mixture model, such that we can model different movement types in the same dataset.

3.1 Introduction

Movement primitives (MPs) are a compact parametric description of a movement [60, 29, 40, 15]. They provide a powerful framework for data driven movement generation as they can be learned from demonstrations as well as by reinforcement learning. They can adapt to a new task by adapting a given set of meta-parameters [79, 42, 50]. For example, the final joint positions or the execution speed [29] of the movement can be adapted.

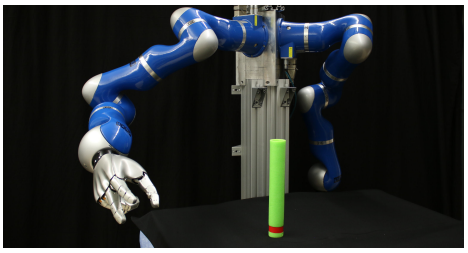


Figure 3.1: The robot used in the experiments to learn trajectory distributions.

Yet, for most movement primitive representations, the set of meta-parameters is pre-coded into the movement primitive representation and can not be adapted. However, for most tasks, a different encoding of the meta-parameters might be more appropriate than the pre-coded parameters of the primitive representation. We believe that this shortcoming has also hindered the application of movement primitives for more complex multi-task learning applications. In this paper we want to learn the encoding of the meta-parameters also from data. Our approach extracts a low-dimensional manifold in the MP parameter space. Each point on this manifold is described by a small set of control variables. Hence, our underlying assumption is that, while the parametrization of movements might be high-dimensional, useful parameter vectors for a given set of tasks typically share a lot of structure, i.e. they lie on a lower dimensional manifold. Each demonstration can now be characterized by the corresponding control variables that can be seen as a compact description of the task considered in this demonstration. For example, in a table tennis scenario, these control variables could specify the location of the hitting point or the desired return direction for the ball. Hence, our model can not only be applied for efficient generalization in multi-task learning with movement primitives but is also well suited for analyzing the movements of human demonstrators. We represent the latent manifold model by a hierarchical Bayesian model. The control variables for each demonstrations are treated as latent variables that are also inferred from the data. The model is extended by a mixture model such that we can learn the control variables of multiple types of movements. We will use Probabilistic Movement Primitives (ProMPs) as underlying movement primitive representation as they can be naturally integrated in the Hierarchical Bayesian Model (HBM) representation. When learning or analyzing movement data, we have to deal with several challenges, such as high-dimensionality, noise, missing data, partial observations, and the data can contain multiple modes that represent different types of movements. In order to deal with all these requirements, we apply a fully Bayesian approach where we integrate out all the estimated parameters of the model. In our experiments, we will illustrate the improved generalization properties of our approach compared to the standard ProMP approach in the case of a small amount of training data and show how demonstrations can be easily analyzed and characterized by the extracted latent control variables.

3.1.1 Related Work

Movement primitives can be categorized into trajectory-based [29, 68, 60] and state-based representations [40]. In this paper we will focus on trajectory based approaches as they are more commonly used and easier to scale up to higher dimensions. A common trajectory-based approach are the dynamical movement primitives (DMPs). DMPs [29] are represented by a parametrized dynamical system that is given by a linear point-attractor that is perturbed by a non-linear time dependent forcing function. The forcing function can be used to encode an arbitrary shape of the trajectory and the weights of the forcing function can be easily obtained from demonstrations by linear regression. One of the benefits of the DMP approach is that it

specifies a small set of meta-parameters. These meta-parameters include the final position of the movement, which is given by the point attractor, the final velocities, the execution speed, or the amplitude of the movement [41, 63, 29]. In multi-task learning with DMPs [42, 25, 50], it is a common strategy to only adapt the meta-parameters due to the high dimensionality of the weights of the forcing function. While DMPs have several more benefits such as stability, and the ability to represent stroke based and rhythmic movements, DMPs also have several limitations, such as that they can not represent optimal behavior in stochastic systems and the adaptation of the trajectory due to the meta-parameters is based on heuristics. These issues have been addressed by the recently proposed Probabilistic Movement Primitives approach [60, 61]. ProMPs estimate a distribution of trajectories instead of encoding single trajectories. The main benefit of the probabilistic representation is that we can use probabilistic operators such as conditioning for adaptation and a product of distribution for co-activating primitives. A distribution over trajectories also contains information on which time points are relevant for the movement, e.g., time points with small variance in the Cartesian end-effector space could denote task relevant via-points or targets. However, in difference to DMPs, ProMPs are lacking meta-parameters that can be used to adapt the trajectories with a small amount of control variables. While it would be easy to pre-specify such control variables by conditioning the trajectory distribution for a fixed set of time points, such an approach would again require a lot of manual tuning and is lacking flexibility.

Our approach automatically extracts a small amount of control variables from a given set of demonstrations in the ProMP framework. We use a hierarchical Bayesian approach to model prior distributions, which is inspired by techniques from multi-task learning (MTL) [84, 52, 49, 71]. In MTL the underlying assumption is that multiple tasks (or trajectories) share a common structure, and, hence, with an increasing number of related tasks that have been already learned, the number of needed training samples for generalizing to a new task decreases [1]. This property is highly desired in robotics, where the data is often high dimensional and obtaining training samples is costly. Different approaches exist to model the shared information across tasks. They can be roughly separated into two different categories, i.e. methods where parameters of the model are close to each other in a geometric sense [23, 71] and approaches where the parameters of the model share a common structure [84, 81, 14, 52, 65, 62]. This structure can be a clustering assumption [81], a (Gaussian) prior for the parameters of all tasks [84, 52] or some advanced structure like the Kingman's coalescent [14], which is a continuous time, partitioned valued Markov process. Our approach is highly related to the Bayesian MTL approach presented in [62], where a prior distribution over parameters is learned. The prior distribution is assumed to have a low-dimensional, latent structure that is represented by a linear factor model. In order to represent several modes (or non-linearities) in the data, the model is extended to a mixture model of linear factor models. For both, the number of mixture components and the number of factors, a non-parametric Dirichlet prior has been used. All parameters of the model are integrated out by the use of a combination of sampling and variational inference. We will use a simplification of this model, assuming a fixed number of mixture components, without the Dirichlet priors, allowing a much more efficient algorithm without the need for expensive sampling methods. We extend the model of Passos et al. by an additional hyper-prior and show that this hyper-prior significantly increases the robustness of the Bayesian model.

3.2 Probabilistic Movement Primitives

In this section we will give a brief overview on Probabilistic Movement Primitives (ProMPs) as they provide the foundation for our hierarchical Bayesian model. ProMPs represent a movement by a distribution $p(\boldsymbol{\tau})$ over trajectories $\boldsymbol{\tau} = \boldsymbol{y}_{1:T}$, where $\boldsymbol{y}_t$ specifies the joint positions (or any other quantities, such as a Cartesian coordinates of a ball) at time step t. ProMPs use a linear basis function model with J basis functions to represent a single trajectory, i.e.

$$p(\boldsymbol{y}_t|\boldsymbol{w}) = \mathcal{N}\left(\boldsymbol{y}_t\middle|\boldsymbol{\Psi}_t\boldsymbol{w}, \beta^{-1}\boldsymbol{I}\right) \text{ and } p(\boldsymbol{\tau}) = \prod_{t=1}^{T} p\left(\boldsymbol{y}_t|\boldsymbol{w}\right),$$

where β denotes the precision of the data. The weight vector $\boldsymbol{w}$ is a compact representation of the trajectory. The basis functions $\boldsymbol{\Psi}_t$ only depend on the time or, alternatively, on the phase of the movement. For a single Degree of Freedom (DoF), $\boldsymbol{\Psi}_t$ is just given by a vector of normalized Gaussian basis functions $\boldsymbol{\phi}_t$ with

$$\phi_{t,i} = \frac{\exp\left(-0.5(t-c_i)^2\right)}{\sum_{j=1}^{J}\exp\left(-0.5(t-c_j)^2\right)},$$

where c_i denotes the center of the ith basis function (note that to enhance readability we skipped the bandwidth parameters in this notation).

For multi-dimensional systems with D DoFs, the basis function matrix is represented by a block-diagonal matrix, i.e,

$$\boldsymbol{\Psi}_t = \begin{bmatrix} \boldsymbol{\phi}_t^T & \boldsymbol{0}^T & \dots & \boldsymbol{0}^T \\ \boldsymbol{0}^T & \boldsymbol{\phi}_t^T & \dots & \boldsymbol{0}^T \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{\phi}_t^T \end{bmatrix}.$$

Due to this encoding of the basis function matrix, the trajectories of all DoFs can still be represented as a single weight vector $\boldsymbol{w}^T = [\boldsymbol{w}_1^T, \boldsymbol{w}_2^T, \dots, \boldsymbol{w}_D^T]$ that is given by a concatenation of all weight vectors for each degree of freedom.

Still, a single weight vector $\boldsymbol{w}$ only represents a single trajectory $\boldsymbol{\tau}$. In order to represent a distribution over trajectories $p(\boldsymbol{\tau})$, we can estimate a distribution $p(\boldsymbol{w})$ over the weight vectors and, subsequently, integrate out the weight vectors. In the original ProMP approach, a multivariate Gaussian distribution is used to model the prior distribution

$$p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w). \tag{3.1}$$

As such, the distribution over trajectories is also Gaussian and can be computed in closed form

$$\begin{aligned} p(\boldsymbol{\tau}) &= \int p(\boldsymbol{\tau}|\boldsymbol{w})p(\boldsymbol{w})d\boldsymbol{w}, \\ &= \int \mathcal{N}\left(\boldsymbol{y}_{1:T}\middle|\boldsymbol{\Psi}_{1:T}\,\boldsymbol{w}, \beta^{-1}\boldsymbol{I}\right)\mathcal{N}\left(\boldsymbol{w}|\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w\right)d\boldsymbol{w}, \\ &= \mathcal{N}\left(\boldsymbol{y}_{1:T}\middle|\boldsymbol{\Psi}_{1:T}\,\boldsymbol{w}, \boldsymbol{\Psi}_{1:T}\boldsymbol{\Sigma}_w\boldsymbol{\Psi}_{1:T}^T + \beta^{-1}\boldsymbol{I}\right), \end{aligned}$$

where $\boldsymbol{\Psi}_{1:T}$ is a $TD \times DJ$ matrix containing the basis function matrices for all time steps and $\boldsymbol{w}$ is a DJ dimensional column vector.

3.2.1 Learning from Demonstrations with ProMPs

A ProMP already defines a simple hierarchical Bayesian model in a similar fashion as a Bayesian linear regression model. The mean $\boldsymbol{\mu}_w$ and the covariance matrix $\boldsymbol{\Sigma}_w$ can be learned from data by maximum likelihood using the Expectation Maximization (EM) algorithm [20]. A simpler solution that works well in practice is to compute first the most likely estimate of $\boldsymbol{w}^{[i]}$ for each trajectory $\boldsymbol{\tau}^{[i]}$ independently, where the index i denotes the i-th demonstration[1]. Subsequently, mean and covariance of $p(\boldsymbol{w})$ can be estimated by the sample mean and sample covariance of the $\boldsymbol{w}^{[i]}$'s. One advantage of the EM based approach in comparison to the more direct approach is that the EM algorithm can also be used for learning from incomplete data where, e.g., some segments of the trajectories might be missing due to occlusions in vision based recordings.

However, the training of ProMPs also suffers from a severe disadvantage. As the model has a lot of parameters due to the high-dimensional covariance matrix, ProMPs suffer from over-fitting if we have little training data and noisy trajectories. The more sophisticated hierarchical Bayesian model for ProMPs introduced in this paper alleviates this problem.

3.2.2 Predictions with ProMPs by Conditioning

ProMPs can also be used to predict the behavior of the demonstrator once we have seen an initial part of a new trajectory. Lets assume that we have observed a human demonstrator at $m = 1, 2, ..., M$ different time points[2] t_1 to t_M at the positions $\boldsymbol{y}_{t_1}$ to $\boldsymbol{y}_{t_M}$. Let us further denote $\boldsymbol{\Psi_o}$ as the concatenation of the basis function matrices for these time points and $\boldsymbol{o}$ as concatenation of the $\boldsymbol{y}_{t_m}$ vectors. Given these observations, we can obtain a conditioned distribution $p(\boldsymbol{w}|\boldsymbol{o})$ over the weight vectors. This distribution is Gaussian with mean and variance

$$\begin{aligned}\boldsymbol{\mu}_{\boldsymbol{w}|\boldsymbol{o}} &= \boldsymbol{\mu}_{\boldsymbol{w}} + \\ &\quad \boldsymbol{\Sigma_w}\boldsymbol{\Psi}_{\boldsymbol{o}}^T \left(\boldsymbol{\Sigma}_o + \boldsymbol{\Psi_o}\boldsymbol{\Sigma_w}\boldsymbol{\Psi}_{\boldsymbol{o}}^T\right)^{-1} \left(\boldsymbol{o} - \boldsymbol{\Psi}_o\boldsymbol{\mu_w}\right), \qquad (3.2)\\ \boldsymbol{\Sigma}_{\boldsymbol{w}|\boldsymbol{o}} &= \boldsymbol{\Sigma_w} - \boldsymbol{\Sigma_w}\boldsymbol{\Psi}_{\boldsymbol{o}}^T \left(\boldsymbol{\Sigma}_o + \boldsymbol{\Psi_o}\boldsymbol{\Sigma_w}\boldsymbol{\Psi}_{\boldsymbol{o}}^T\right)^{-1} \boldsymbol{\Psi_o}\boldsymbol{\Sigma_w}. \qquad (3.3)\end{aligned}$$

The conditional distribution $p(\boldsymbol{w}|\boldsymbol{o})$ can be used to predict the behavior of the demonstrator for future time points $t > t_M$, i.e. we can determine the mean and covariance of $\boldsymbol{y}$ for future time points. Note that the same procedure can be applied for partial observations, where only a subset of the quantities in $\boldsymbol{y}_t$ is observed. The covariance matrix $\boldsymbol{\Sigma}_o$ can be used to control the importance of different dimensions.

3.3 Extracting Control Variables with Hierarchical Priors

Our goal is to model non-linear prior distributions that can be modulated by low-dimensional latent control variables. We define a hierarchical prior on the weight vector $\boldsymbol{w}$ using mixture

[1] Given a trajectory $\boldsymbol{\tau}_i$, the corresponding weight vectors $\boldsymbol{w}^{[i]}$ can be estimated by a straight forward least squares estimate.

[2] Note that these time points do not need to be sampled in uniform time intervals.

models

$$p(\boldsymbol{w}^{[i]}) = \sum_{k=1}^{K} \pi_k \mathcal{N}\left(\boldsymbol{w}^{[i]} \middle| \boldsymbol{b}_k + \boldsymbol{M}_k \boldsymbol{h}_k^{[i]}, \alpha^{-1}\boldsymbol{I}\right). \tag{3.4}$$

The vector $\boldsymbol{b}_k$ denotes an offset term and the projection matrix $\boldsymbol{M}_k$ defines the mapping from the low-dimensional control variables $\boldsymbol{h}_k^{[i]}$ to the weight vector $\boldsymbol{w}^{[i]}$ of trajectory i. The parameter α models the precision of the latent manifold priors and π_k denotes the mixing coefficients. The different mixture components can model different movement types, e.g., forehand and backhand strokes in a table tennis game. Within a mixture component the latent control variable $\boldsymbol{h}_k^{[i]}$ models the adaptation of the movement to the current task.

All parameters of this prior distribution are unknown a priori and are learned from demonstrations. We follow a fully Bayesian approach, where we treat all parameters as random variables and introduce conjugate priors for these random variables. We derive variational update equations for all relevant distributions. We also demonstrate how predictions can be computed by conditioning with the hierarchical priors.

We will start our discussion for the most simple case, using only a single mixture component.

3.3.1 Control Variables for a Single Movement Type

For a single mixture component the prior in Eq. (3.4) simplifies to

$$p(\boldsymbol{w}^{[i]}) = \mathcal{N}\left(\boldsymbol{w}^{[i]} \middle| \boldsymbol{b} + \boldsymbol{M}\boldsymbol{h}^{[i]}, \alpha^{-1}\boldsymbol{I}\right).$$

We introduce conjugate priors for the random variables, i.e. we use $p(\boldsymbol{b}) = \mathcal{N}(\boldsymbol{b}|\boldsymbol{0}, \boldsymbol{I})$ for the offset vector, $p(\boldsymbol{h}^{[i]}) = \mathcal{N}(\boldsymbol{h}^{[i]}|\boldsymbol{0}, \boldsymbol{I})$ for the control variables $\boldsymbol{h}^{[i]}$, $\alpha = \Gamma(\alpha|a_0, b_0)$ for the precision[3] α, and $p(\boldsymbol{M}) = \prod_v \mathcal{N}(\boldsymbol{m}^{[v]}|\boldsymbol{0}, {\lambda^{[v]}}^{-1}\boldsymbol{I})$ for the projection matrix $\boldsymbol{M}$. Here, $\boldsymbol{m}^{[v]}$ denotes the v-th column of the matrix $\boldsymbol{M} = [\boldsymbol{m}^{[1]}, \boldsymbol{m}^{[2]}, \ldots, \boldsymbol{m}^{[V]}]$, with V denoting the dimensionality of the latent variable $\boldsymbol{h}^{[i]}$. The symbol Γ denotes the Gamma distribution.

To enhance the numerical stability of the variational updates, we also add a gamma prior on the precision parameters of the projection matrix, i.e. $p(\lambda^{[v]}) = \Gamma(\lambda^{[v]}|c_0, d_0)$. The influence of this additional prior is evaluated in the experimental section.

As we use a variational inference approach [3], we assume a complete factorization of the variational posterior given by

$$q(\boldsymbol{\xi}) = q(\boldsymbol{b})q(\boldsymbol{M})q(\lambda_{1:V})q(\alpha)\prod_{i=1}^{L} q(\boldsymbol{w}^{[i]})q(\boldsymbol{h}^{[i]}),$$

where $\boldsymbol{\xi} = \{\boldsymbol{w}^{[1:L]}, \boldsymbol{h}^{[1:L]}, \boldsymbol{b}, \boldsymbol{M}, \lambda_{1:V}, \alpha\}$ and L denotes the total number of demonstrations. The variational distributions for the weight vector $\boldsymbol{w}^{[i]}$, the latent variable $\boldsymbol{h}^{[i]}$, the offset vector $\boldsymbol{b}$, the v-th column of the projection matrix $\boldsymbol{M}$, are specified as $q(\boldsymbol{w}^{[i]}) := \mathcal{N}(\boldsymbol{w}^{[i]}|\boldsymbol{\mu}_{\boldsymbol{w}^{[i]}}, \boldsymbol{\Sigma}_{\boldsymbol{w}^{[i]}})$, $q(\boldsymbol{h}^{[i]}) := \mathcal{N}(\boldsymbol{h}^{[i]}|\boldsymbol{\mu}_{\boldsymbol{h}^{[i]}}, \boldsymbol{\Sigma}_{\boldsymbol{h}^{[i]}})$, $q(\boldsymbol{b}) := \mathcal{N}(\boldsymbol{b}|\boldsymbol{\mu}_{\boldsymbol{b}}, \sigma_{\boldsymbol{b}}\boldsymbol{I})$, and $q(\boldsymbol{m}^{[v]}) := \mathcal{N}(\boldsymbol{m}^{[v]}|\boldsymbol{\mu}_{\boldsymbol{m}^{[v]}}, \sigma_{\boldsymbol{m}^{[v]}}\boldsymbol{I})$. The remaining definitions are listed in the appendix.

[3] To make this prior non-informative we use $a_0 = 1e-5$ and $b_0 = 1e-5$.

The most important variational update equations read

$$\begin{aligned}
\boldsymbol{\mu}_{\boldsymbol{w}^{[i]}} &= \boldsymbol{\Sigma}_{\boldsymbol{w}^{[i]}} \left(\beta {\boldsymbol{\Psi}^{[i]}_{1:T}}^T \boldsymbol{y}^{[i]}_{1:T} + \bar{\alpha} \left(\boldsymbol{\mu}_{\boldsymbol{b}} + \bar{\boldsymbol{M}} \boldsymbol{\mu}_{\boldsymbol{h}^{[i]}} \right) \right), \\
\boldsymbol{\Sigma}_{\boldsymbol{w}^{[i]}} &= \left(\beta {\boldsymbol{\Psi}^{[i]}_{1:T}}^T \boldsymbol{\Psi}^{[i]}_{1:T} + \bar{\alpha} \boldsymbol{I} \right)^{-1}, \\
\boldsymbol{\mu}_{\boldsymbol{h}^{[i]}} &= \bar{\alpha}\, \boldsymbol{\Sigma}_{\boldsymbol{h}^{[i]}}\, \bar{\boldsymbol{M}}^T \left(\boldsymbol{\mu}_{\boldsymbol{w}^{[i]}} - \boldsymbol{\mu}_{\boldsymbol{b}} \right), \\
\boldsymbol{\mu}_{\boldsymbol{b}} &= \sigma_{\boldsymbol{b}} \bar{\alpha} \boldsymbol{I} \left(\sum_{i=1}^{L} \left(\boldsymbol{\mu}_{\boldsymbol{w}^{[i]}} - \bar{\boldsymbol{M}} \boldsymbol{\mu}_{\boldsymbol{h}^{[i]}} \right) \right), \\
\boldsymbol{\mu}_{\boldsymbol{m}^{[v]}} &= \sigma_{\boldsymbol{m}^{[v]}} \bar{\alpha} \boldsymbol{I} \left(\sum_{i=1}^{L} \mu_{h^{[v,i]}} \left(\boldsymbol{\mu}_{\boldsymbol{w}^{[i]}} - \boldsymbol{\mu}_{\boldsymbol{b}} \right) \right),
\end{aligned}$$

where $\bar{\boldsymbol{M}} = [\boldsymbol{\mu}_{\boldsymbol{m}^{[1]}}, \ldots, \boldsymbol{\mu}_{\boldsymbol{m}^{[V]}}]$. The inferred feature precision is denoted by $\bar{\alpha}$ and the scalar $\mu_{h^{[v,i]}}$ denotes the v-th element in the vector $\boldsymbol{\mu}_{\boldsymbol{h}^{[i]}} = \left[\mu_{h^{[1,i]}}, \ldots, \mu_{h^{[V,i]}}\right]^T$.

Compared to the prior used in ProMPs in Eq. (3.1), the combination of the latent variable $\boldsymbol{\mu}_{\boldsymbol{h}^{[i]}}$ and the projection matrix $\bar{\boldsymbol{M}}$ implements a more accurate model of the prior distribution. As we will demonstrate, this hierarchical prior model is less sensitive to overfitting in the case of noisy observations or incomplete data.

3.3.2 Predictions by Conditioning the Hierarchical Prior

In the hierarchical prior model, predictions are performed by computing the conditioned distribution over the latent task variable $p(\boldsymbol{h}|\boldsymbol{o})$. This conditioned distribution can be simply determined by integrating out the weight vector $\boldsymbol{w}$

$$\begin{aligned}
p(\boldsymbol{h}|\boldsymbol{o}) &\propto p(\boldsymbol{o}|\boldsymbol{h}) p(\boldsymbol{h}), \\
&= \int_{\boldsymbol{w}} p\left(\boldsymbol{o}|\boldsymbol{\Psi}_{\boldsymbol{o}}, \boldsymbol{w}\right) p\left(\boldsymbol{w}|\boldsymbol{h}\right) p(\boldsymbol{h}) d\boldsymbol{w}, \\
&= \mathcal{N}\left(\boldsymbol{o} \middle| \boldsymbol{\Psi}_{\boldsymbol{o}} \left(\boldsymbol{\mu}_{\boldsymbol{b}} + \bar{\boldsymbol{M}} \boldsymbol{h} \right), \boldsymbol{\Sigma}_o + \bar{\alpha}^{-1} \boldsymbol{\Psi}_{\boldsymbol{o}} \boldsymbol{\Psi}_{\boldsymbol{o}}^T \right) p(\boldsymbol{h}),
\end{aligned}$$

where $p(\boldsymbol{h})$ is the Gaussian prior distribution for the latent variable. Now, we can condition on the control variable $\boldsymbol{h}$ on the demonstrations to obtain a Gaussian over $\boldsymbol{h}$ with mean and variance

$$\boldsymbol{\mu}_{\boldsymbol{h}|\boldsymbol{o}} = \bar{\boldsymbol{M}}^T \boldsymbol{\Psi}_{\boldsymbol{o}}^T \boldsymbol{A}^{-1} \left(\boldsymbol{o} - \boldsymbol{\Psi}_{\boldsymbol{o}} \boldsymbol{\mu}_{\boldsymbol{b}} \right), \tag{3.5}$$

$$\boldsymbol{\Sigma}_{\boldsymbol{h}|\boldsymbol{o}} = \boldsymbol{I} - \bar{\boldsymbol{M}}^T \boldsymbol{\Psi}_{\boldsymbol{o}}^T \boldsymbol{A}^{-1} \boldsymbol{\Psi}_{\boldsymbol{o}} \bar{\boldsymbol{M}}, \tag{3.6}$$

where $\boldsymbol{A} = \boldsymbol{\Sigma}_o + \boldsymbol{\Psi}_{\boldsymbol{o}} \left(\bar{\alpha}^{-1} \boldsymbol{I} + \bar{\boldsymbol{M}} \bar{\boldsymbol{M}}^T \right) \boldsymbol{\Psi}_{\boldsymbol{o}}^T$.

Given the distribution over the inferred latent task variable the posterior over feature weights is given by

$$\boldsymbol{\mu}_{\boldsymbol{w}|\boldsymbol{o}} = \boldsymbol{\mu}_{\boldsymbol{b}} + \bar{\boldsymbol{M}} \boldsymbol{\mu}_{\boldsymbol{h}|\boldsymbol{o}}, \tag{3.7}$$

$$\boldsymbol{\Sigma}_{\boldsymbol{w}|\boldsymbol{o}} = \bar{\alpha}^{-1} \boldsymbol{I} + \bar{\boldsymbol{M}} \boldsymbol{\Sigma}_{\boldsymbol{h}|\boldsymbol{o}} \bar{\boldsymbol{M}}^T. \tag{3.8}$$

It is illustrative to investigate the differences of the standard conditioning of the ProMPs in Eq. (3.2) and Eq. (3.3) to the conditioning with the hierarchical prior. The conditioning in the ProMP case requires a full-rank covariance matrix, which is hard to obtain given a small amount of training data. In contrast, the latent prior model only requires the projection matrix $\bar{\boldsymbol{M}}$ to perform the conditioning. Hence, the predictions of the latent prior model are less prone to overfitting and are, therefore, also applicable for a small amount of training data.

3.3.3 Extension to Multiple Movement Types ($K > 1$)

The mixture distribution in Eq. (3.4) adds an additional multinomial variable per demonstration to our probabilistic model, i.e. $z_k^{[i]} \in \{0, 1\}$. We represent this multinomial variable as binary vector $\mathbf{z}^{[i]} = \{z_1^{[i]}, ..., z_K^{[i]}\}$.

To derive variational updates, we specify a multinomial hyper-prior for the mixing indices $p(\boldsymbol{Z}) = \prod_{i=1}^{L} \prod_{k=1}^{K} (\boldsymbol{\pi}_k)^{z_k^{[i]}}$.

The variational updates are the same as for the case with only a single component, with the difference that the trajectories are weighted by the responsibilities of the individual mixture components $\mu_{z_k^{[i]}}$, i.e.

$$\begin{aligned}
\boldsymbol{\mu}_{\boldsymbol{w}^{[i]}} =& \boldsymbol{\Sigma}_{\boldsymbol{w}^{[i]}} \Bigg(\beta {\boldsymbol{\Psi}_{1:T}^{[i]}}^T \boldsymbol{y}_{1:T}^{[i]} + \\
& \sum_{k=1}^{K} \bar{\alpha}_k \mu_{z_k^{[i]}} \left(\boldsymbol{\mu}_{\boldsymbol{b}_k} + \bar{\boldsymbol{M}}_k \boldsymbol{\mu}_{\boldsymbol{h}_k^{[i]}} \right) \Bigg), \\
\boldsymbol{\Sigma}_{\boldsymbol{w}^{[i]}} =& \left(\beta {\boldsymbol{\Psi}_{1:T}^{[i]}}^T \boldsymbol{\Psi}_{1:T}^{[i]} + \sum_{k=1}^{K} \bar{\alpha}_k \mu_{z_k^{[i]}} \boldsymbol{I} \right)^{-1}.
\end{aligned}$$

Computing predictions with the mixture model is also straight forward. For each component we compute the conditioned distribution on the latent control variables as in Eq. (3.5) and in Eq. (3.6) and the posterior over the feature weights using Eq. (3.7) and Eq. (3.8). Thereafter the posterior distributions are weighted by the responsibilities of each mixture model

$$\begin{aligned}
z^{[k]} &= \frac{\pi_k \mathcal{N}\left(\boldsymbol{o} \middle| \boldsymbol{\mu}_{\boldsymbol{w}|\boldsymbol{o}}^{[k]}, \boldsymbol{\Sigma}_{\boldsymbol{w}|\boldsymbol{o}}^{[k]} \right)}{\sum_{j=1}^{K} \pi_j \mathcal{N}\left(\boldsymbol{o} \middle| \boldsymbol{\mu}_{\boldsymbol{w}|\boldsymbol{o}}^{[j]}, \boldsymbol{\Sigma}_{\boldsymbol{w}|\boldsymbol{o}}^{[j]} \right)}, \\
\boldsymbol{\Sigma}_{\boldsymbol{w}|\boldsymbol{o}} &= \sum_{k=1}^{K} z^{[k]} \boldsymbol{\Sigma}_{\boldsymbol{w}|\boldsymbol{o}}^{[k]}, \\
\boldsymbol{\mu}_{\boldsymbol{w}|\boldsymbol{o}} &= \sum_{k=1}^{K} z^{[k]} \boldsymbol{\mu}_{\boldsymbol{w}|\boldsymbol{o}}^{[k]}.
\end{aligned}$$

The remaining updates are listed in the appendix.

3.4 Results

We evaluate our method on two real robot tasks. In the first task the robot played a table tennis game and we recorded the Cartesian coordinates of a racket mounted at its end-effector and the Cartesian coordinates of the ball. A Barrett WAM anthropomorphic arm was used for this experiment [54]. The robot provides regular updates about its joint positions at a rate of 1KHz that are used by the forward kinematics to compute the Cartesian position of the racket. The ball is tracked by a high-speed, multi-camera vision system [51] that provides updates at a rate of 200Hz. The extracted dataset contains twenty ball and racket trajectories.

In the second task we placed an obstacle in front of a KUKA lightweight arm and demonstrated by kinesthetic teaching different ways to approach a desired target point in Cartesian space. During the demonstrations we avoided hitting the obstacle and we bypassed it either by moving to the left or to the right. The demonstrations are depicted in Fig. 3.5. For this experiment we recored the Cartesian position and orientation of the end-effector. The state vector $\boldsymbol{y}_t$ for this experiment is seven dimensional, three dimensions for the position and four for the quaternion based orientation.

3.4.1 Summary of the investigated features

We compare the proposed model, denoted as Latent Manifold ProMPs (LMProMPs) in the figures, to the standard ProMP approach in the two robotic setups.

In the table tennis scenario we investigate the effect of noise and missing data on predicting the final ball impact location at the opponent's side of the table and we demonstrate how the learned latent variables can be used to semantically analyze the data.

Additionally, we demonstrate the beneficial properties of the mixture model in representing the bi-modal distribution required to successfully execute the KUKA reaching task. We use the learned mixture model to generate trajectories to new target locations, not encountered during training, and execute them on the real robot. We demonstrate that our proposed approach successfully avoids the obstacle, while the standard ProMPs average over the two modes and the generalization fails.

In both experiments we used linear regression to compute the feature weights $\boldsymbol{w}$ and we subsequently applied a principal component analysis. We initialized our model with the first ten principal components.

3.4.2 The effect of noise and missing data

We use the table tennis setup to predict the final impact location of the ball at the opponent's court. We evaluate our prediction by computing the Euclidean distance in the x,y-plane to the true impact location. The dataset used for learning is shown in Fig. 3.2(A-B). It should be noted that the colors (red and blue) in Fig. 3.2 are only used for the visualization as no labels were used for modeling the data.

For a baseline comparison we trained the ProMPs on the same data. The learned distributions over trajectories for ProMPs are illustrated for three Cartesian coordinates in Fig. 3.2(C-E). We denote the mean of the trajectory distribution with a solid black line and the standard deviation by the shaded region.

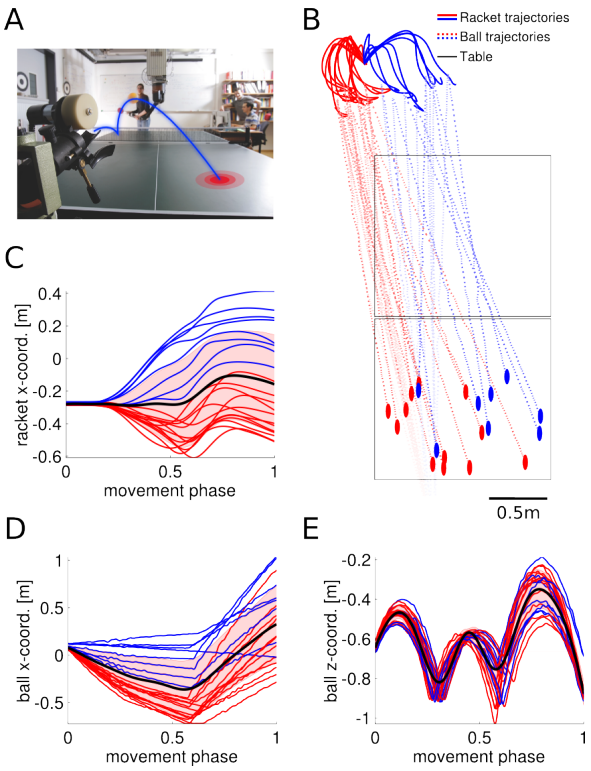


Figure 3.2: (A-B) Trajectory prediction task in a table tennis setting using 20 end-effector and ball trajectories. (C-E) Learned distributions over trajectories for three dimensions (out of six) using ProMPs. The colors (red and blue) are only used to visualize differences in the movement directions.

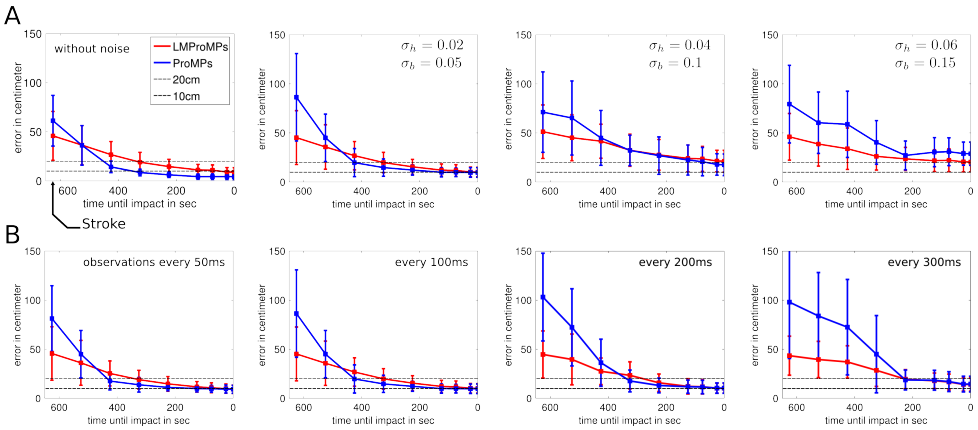


Figure 3.3: The effect of noise (A) and missing data (B) on the prediction performance of ProMPs (blue lines) and LM-ProMPs (red lines). In (A), from left to right the amount of applied noise is increased. In (B) four different frame rates of observations ($\in \{50, 100, 200,$ and $300\}$ms) are investigated.

In the collected dataset, the robot returns the ball within 550ms to 650ms in advance to the final ball impact. In our comparison, we analyze the prediction performance with respect to the time until the impact event, where we focus on the movement phase right after the stroke, ≈ 625ms before the end. We used leave-one-out cross-validation to compute the test error.

A fast multi-camera vision setup, good lighting conditions, and access to the opponents sensor readings are amenities we can not always afford. Therefore, we simulate the effect of noisy and incomplete observations, and we evaluate their impact on the prediction performance. First, we add zero-mean Gaussian observation noise to the Cartesian coordinates of the racket and to the Cartesian coordinates of the ball. The standard deviation of the noise used in our evaluation is $\sigma_h \in 10^{-2}\{0, 2, 4, 6\}$ and $\sigma_b \in 10^{-2}\{0, 5, 10, 15\}$ for the racket and the ball, respectively. The results are illustrated in Fig. 3.3(A), where we show the advantage of the learned prior distribution using latent variables.

Additionally, we evaluate the effect of sparse observations using different sampling intervals, $\{50, 100, 200,$ and $, 300\}$ms. The proposed model is more robust with respect to sparse observations, whereas the standard ProMPs overfit to the training data, especially in the early phase of the movement. The performance comparison of the two approaches is illustrated in Fig. 3.3(B).

3.4.3 Analyzing the model parameters

As opposed to most movement primitive approaches, our model has only one free parameter to choose that is the precision of the data denoted by β. For large β values the number of contributing latent variables in the generative model is increased, and, at some point, the model will overfit to the training data. To analyze this effect, we approximate the complexity of the learned model by computing the rank of the linear feature weights denoted by $\boldsymbol{Mh}^{[i]}$ in Eq. (3.4).

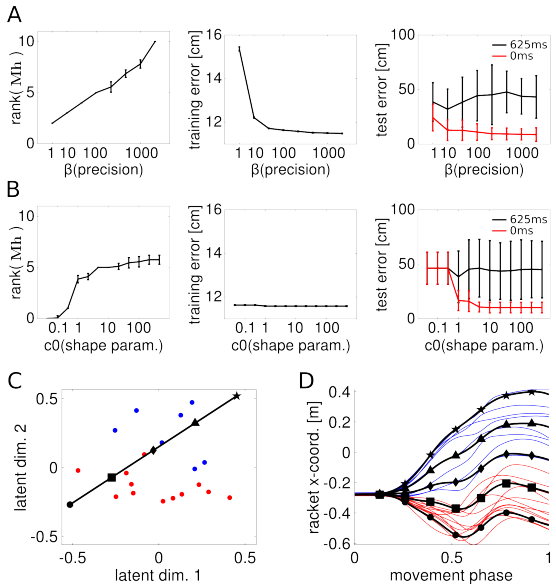


Figure 3.4: (A) The data precision parameter β can be used to adapt the model complexity while avoiding overfitting (shown in the 2nd and 3rd panel for two planning horizons until the ball impact). (B) The gamma prior on the precision parameters λ to increase the numerical stability has little effect on the prediction performance (for $c_0 \geq 1$). (C) Investigation of the effect of the latent variables, where the first dimension of **h** describes the slope whereas the second dimension relates to the waviness (D).

For values of $\beta \in \{1, 10, 50, 100, 200, 500, 1000, 5000\}$ we compute the training and test error. The prediction performance is shown in Fig. 3.4(A). The lowest test error was achieved for $\beta = 10$ (for a prediction horizon of 625ms). Note that the test error will not converge to zero due to noise introduced with $\sigma_h = 0.02$ and $\sigma_b = 0.05$, and the sparse observations at 50ms intervals.

The numerical stability of the LMProMPs can be increased with the addition of a gamma prior on the $\lambda^{[v]}$ parameters, discussed in Subsection 3.3.1. To investigate the influence of this regularization on the test error, we evaluated gamma priors with a constant mean ($c_0/d_0 = 100$) and increasing precision in the interval $c_0 \in [0.05, 500]$. For small values of c_0 the prior converges to a uniform distribution. For $c_0 \geq 1$ the variational updates were numerically stable and the gamma prior had only little influence on the test error, as shown in Fig. 3.4(B).

Finally, we semantically analyze the table tennis dataset to evaluate how the latent variable affect the learned prior distribution. We trained the model with 10-dimensional latent variables $\boldsymbol{h}^{[i]}$ in Eq. (3.4). The effect of the first two latent dimensions in the generative model is illustrated in Fig. 3.4(C-D). The two latent dimensions of the model affect the slope and the waviness of the x-coordinate of the racket trajectories shown in Fig. 3.4(D).

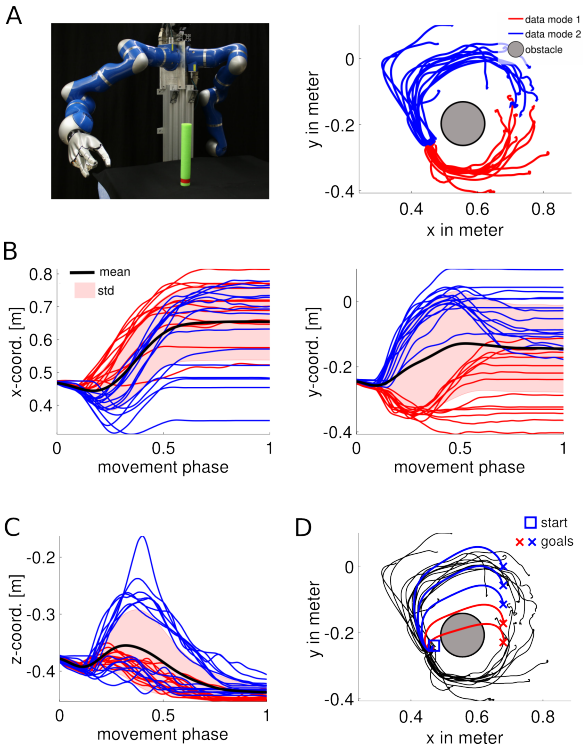


Figure 3.5: (A) Experimental setting and two dimensions out of the 7-dimensional dataset (three end-effector coordinates and the four dimensional quaternions). The colors (red and blue) denote the movement direction to avoid the obstacle. (B-C) Learned distributions using ProMPs. The mean is denoted by the black line and the standard deviation by the shaded region. ProMPs cannot represent the bi-modal distribution in the 2nd panel in (B) and the conditioning on unseen targets might fail (D).

3.4.4 Learning bi-modal trajectory distributions

To demonstrate that LMProMPs can model multi-modal distributions, we study demonstrations of a bi-modal target-reaching task. A KUKA lightweight arm was used to reach for different target locations on a table while avoiding an obstacle. We used kinesthetic teaching and we demonstrated two different ways to approach the target. The setup and demonstrations are shown in Fig. 3.5(A).

For a comparison, we trained ProMPs to learn from the demonstrations, which were unable to represent the two modes. As a result, generalization by conditioning to not encountered target locations may result in trajectories that pass through the obstacle. The learned distributions and example trajectories are shown in Fig. 3.5(B-C).

In contrast, the LMProMPs model is able to capture the two modes of the demonstrations, as shown in Fig. 3.6. We initialized the experiment with *K-means* clustering method using two components. The learned prior distribution and the influence of the first two dimensions of the latent variable are illustrated in Fig. 3.6(A-B). Each mixture component specializes on one mode of the data. Using the learned bi-modal prior distribution, our model is able to generate trajectories to new target locations that avoid the obstacle as shown in Fig. 3.6(C).

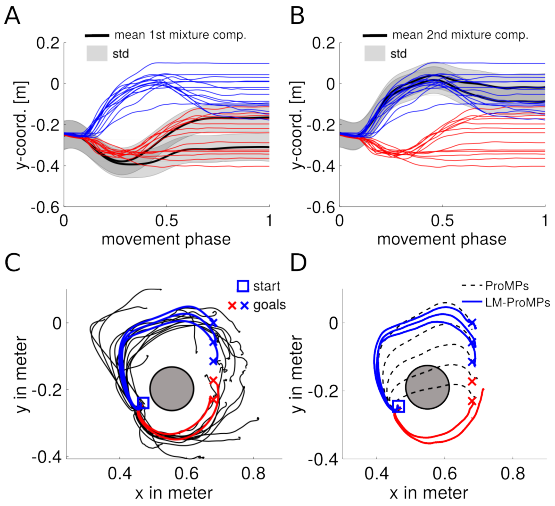


Figure 3.6: Learned bi-modal distribution (the colors red and blue denote the modes) using the proposed mixture model with two mixture components (A-B). The latent variable is used to specialize on subregions within the distribution of the mixture component. This is illustrated for two dimensions of $\mathbf{h}$, where solid black lines denote the mean. (C) Conditioning result using LMProMPs. (D) Real robot results.

The inferred trajectories are smooth and can be executed on the real robot using inverse kinematics to obtain a reference joint trajectory and inverse dynamics control to execute it. The resulting trajectories of the end-effector of the real robot are illustrated in Fig. 3.6(D).

3.5 Conclusion

A desired feature of motor control approaches is to have a low number of control parameters that can be used to adapt learned skills to new or changing situations. In existing movement primitive approaches [60, 29, 40] these control parameters are predefined and can not adapt to the complexity of the tasks. In this paper we proposed a probabilistic movement primitive representation with hierarchical priors that learns these control parameters as well as distributions over trajectories from demonstrations. We demonstrated on two kinesthetic teaching datasets that the control variables can be used to generate new trajectories or to analyze the data. The model naturally extends to mixture models, where multi-modal distributions can be represented. In future work we will investigate non-parametric variants using, e.g., Dirichlet processes on more challenging simulated and real-robot tasks with a larger number of modes.

Chapter 4

Predicting Object Interactions from Contact Distributions (TUD)

Abstract

Contacts between objects play an important role in manipulation tasks. Depending on the locations of contacts, different manipulations or interactions can be performed with the object. By observing the contacts between two objects, a robot can learn to detect potential interactions between them.

Rather than defining a set of features for modeling the contact distributions, we propose a kernel-based approach. The contact points are first modeled using a Gaussian distribution. The similarity between these distributions is computed using a kernel function. The contact distributions are then classified using kernel logistic regression. The proposed approach was used to predict stable grasps of an elongated object, as well as to construct towers out of assorted toy blocks.

4.1 Introduction

Manipulation tasks almost always involve direct physical contact between two or more objects. These contacts can be between different objects in the robot's environment, or between an object and the robot. Depending on the locations of the contacts, different types of interactions and manipulations can occur. For example, a contact on the side of an object may allow for pushing and sliding the object, while a contact on the bottom can be used for lifting or supporting the object. In order to successfully perform a manipulation task, a robot must be able to determine the potential interactions between objects and utilize them to accomplish the task's goal.

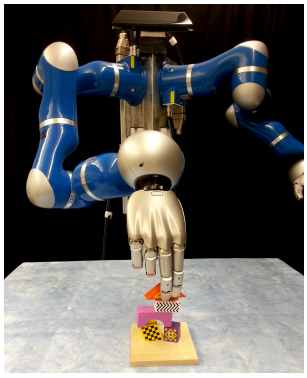


Figure 4.1: The Darias robot performing a block stacking task.

Utilizing contact information in an efficient manner is however not a trivial task. Analytical approaches tend to require accurate models of the objects, and rely on simplified contact models [6]. In an effort to make robots more autonomous, learning approaches

3-Fingered Grasp

4-Fingered Grasp

Figure 4.2: The two types of grasps that were used during the lifting experiment. The three-fingered grasp uses the tips of the thumb, middle, and index fingers in order to pinch the object. The ring and little finger are not touching the box. The four-fingered grasp additionally uses the back of the ring finger on the top of the box in order to provide additional support.

have become more widely adopted in the field of robot manipulation [37, 43, 75]. However, representing contacts between objects often relies on hand-crafted features for the given task.

In this paper, we propose an example-based learning approach to detect interactions between objects from their contact distributions. We pose the problem of detecting interactions as a binary classification problem, wherein the robot has to predict whether or not a certain interaction is occurring based on the geometry and relative poses of the objects. The robot first computes which regions of the objects are in contact with each other. The resulting cloud of contact points is subsequently modeled as a Gaussian distribution. A Bhattacharyya kernel function [34] can then be used to compute the similarities between the contact distributions and, thus, classify them using kernel logistic regression. In this manner, the robot uses the similarity between the current contact distribution and previous distributions in order to classify the potential interaction. The details of the approach are explained in Section 4.2.

The proposed approach was implemented on the real robot shown in Fig. 4.1. In the first experiment, the robot was given the task of predicting which grasps allow it to steadily pick up an elongated object. The second experiment required the robot to stack assorted blocks. The details of the experiments are given in Section 4.3.

4.2 Learning From Contact Distributions

In this section, we first outline related work in interaction detection. In Sections 4.2.2 to 4.2.4, we explain how contacts between objects are detected and used to create contact distributions. In Sections 4.2.5 to 4.2.8, we provide a kernel function for computing the similarity between contact distributions and explain how it is used to classify the distributions using kernel logistic regression.

4.2.1 Related Work

Learning symbolic representations of geometric relations between objects, e.g. object A is ON object B, is an important skill for performing complex manipulation tasks. Rosman and Ramamoorthy [67] proposed the use of a contact network to learn the spatial relations between objects. Contact points were detected using a support vector machine to separate the point clouds of the objects. The vectors between the objects' contact points were then computed and used to classify relations such as *on* and *adjacent* using a k-nearest neighbors classifier. Kulick et al. use an active learning approach to efficiently learn a symbolic representation of the relations between objects [48]. Using features such as the heights of objects and the relative positions between objects, they train a Gaussian process classifier to learn in which geometric states the predicate is true.

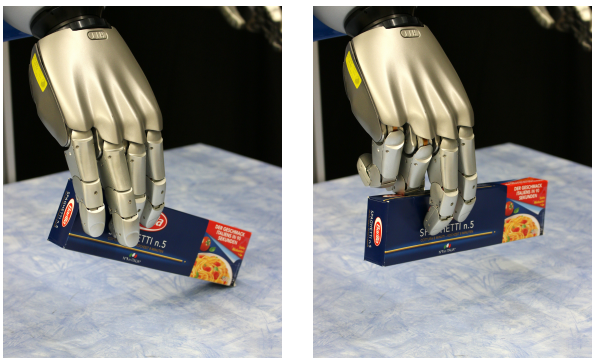
Classifying interactions between objects is also closely related to learning affordances [27]. If an object allows a robot to perform an action with it, than the object is said to "afford" that action. Affordances have been widely studied in robotics [72, 44, 53], and especially in the field of robot grasp synthesis [6]. Recently, several papers have proposed template-based approaches for detecting where an object can be grasped [28, 21, 46]. These approaches predict where to grasp an object based on the local shape of the object relative to the hand. The approach presented by Detry et al. [21] learns both the bounding box of points to consider when comparing grasps as well as a dictionary of graspable parts.

Contact information can also be represented in the form of tactile sensor readings. Bekiroglu et al. [2] proposed learning to predict stable grasps of objects using kernel logistic regression. Their approach used a product of three separate kernels based on the position of the hand relative to the object, the approach direction of the hand, and moment features of the tactile sensor arrays' readings. In the work of Dang et al. [13], the locations of the sensed contact points are defined relative to the palm, and modeled using a bag-of-words representation. A support vector machine is then trained to classify stable and unstable grasps.

The features used by learning algorithms can also be designed to capture specific aspects of the contacts between objects. In [75], a classifier was trained on simulated data to predict interactions, such as support and location control, between pairs of objects. The classifier was provided with 93 features, such as the total contact patch area, and the vector between the closest contact point and the other object. Automatic relevance determination was then used to effectively select a subset of these features. Jiang et al. [37] addressed the problem of learning to place objects in a scene. The placement of an object was represented by a set of 145 features, including features for modeling supporting contacts and the caging of objects. A support vector machine with a shared sparsity structure was then used to classify good and bad placements of objects.

4.2.2 Contact Points

In order to determine the contacts between objects, we first need a suitable representation of the object and its geometry. Given an object O_i, where i specifies the index of the object, we define its geometry as a point cloud with n_i points at positions $\boldsymbol{p}_{ij}$ and corresponding normals $\boldsymbol{u}_{ij}$ for $j \in \{1, \ldots, n_i\}$. Point clouds are flexible object representations that are widely used in robotics [70]. The normals of the points are straightforward to compute using the covariance



Failed Lift Successful Lift

Figure 4.3: Examples of failed and successful lifts. A lift was considered a failure if the object was still touching the table at the end of the trial.

of nearby points and the viewing direction.

The point cloud defines the surface of the object and, hence, also where contacts can potentially be made with another object. In order to obtain a set of contact points, each point in the point cloud is classified as either being in contact with the other object or not. In our experiments, we used logistic regression to classify the points, although other methods for detecting contacts are also applicable. The probability of a point $\boldsymbol{p}_{ic}$ being in contact with the object O_j is given by

$$p(\text{contact}|\boldsymbol{p}_{ic}, \boldsymbol{u}_{ic}, O_j) = \left(1 + \exp\left(\boldsymbol{\phi}^T \boldsymbol{\rho}\right)\right)^{-1},$$

where $\boldsymbol{\phi}$ is a vector of feature functions and $\boldsymbol{\rho}$ is a vector of corresponding weights. We used three features, including a density estimation

$$\phi_1(\boldsymbol{p}_{ic}, O_j) = \sum_k \exp\left(-\frac{\left\|\boldsymbol{p}_{ic} - \boldsymbol{p}_{jk}\right\|^2}{\sigma^2}\right)$$

and a surface normal density estimation

$$\phi_2(\boldsymbol{p}_{ic}, \boldsymbol{u}_{ic}, O_j) = \sum_k (\boldsymbol{u}_{ic}^T \boldsymbol{u}_{jk}) \exp\left(-\frac{\left\|\boldsymbol{p}_{ic} - \boldsymbol{p}_{jk}\right\|^2}{\sigma^2}\right)$$

where σ is the length scale of the density. We also include a bias term $\phi_3 = 1$.

These three features are well-suited for detecting arbitrary contacts between two objects. Some interactions however require specific types of contacts, e.g., cutting requires contact with a sharp edge. The set of features can be easily extended for more specific types of contacts.

Computing a set of weights $\boldsymbol{\rho}$ that maximizes the likelihood of the training data is a convex optimization problem, and can be solved using iterative reweighted least squares, as explained in [4]. A point is classified as a contact point if the probability of contact is greater than 0.5.

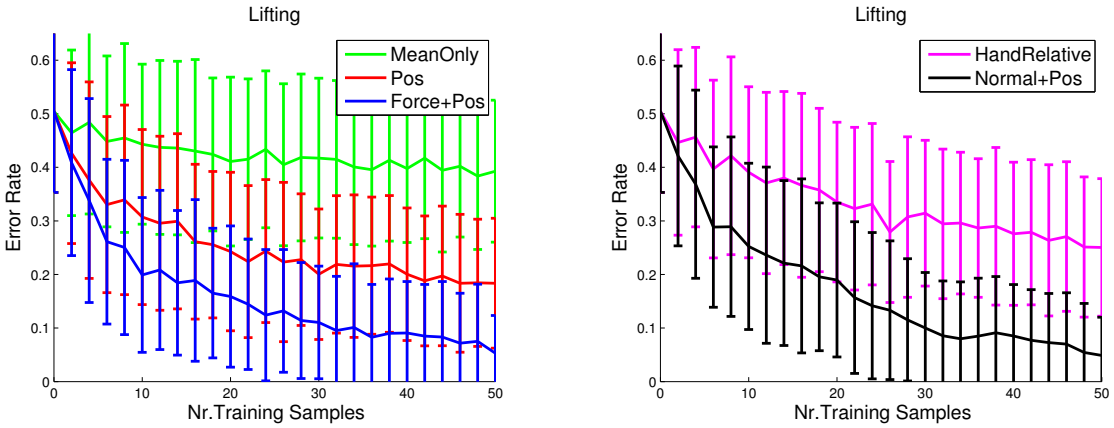


Figure 4.4: The expected error rates for the lifting task. The error bars indicate one standard deviation. An error rate of 1 indicates that none of the test samples were correctly classified, and an error rate of 0 is achieved when the classifier evaluates all of the samples correctly.

4.2.3 Object Centers

In addition to the shape of the object, we also define a set of *object centers* for each object. Object centers are used to define interaction-relevant coordinate frames for the object. Each center $\boldsymbol{c}_{ik}$, where k is the index of the center for object O_i, is associated with a position $\boldsymbol{x}_{ik}$ and at least one axis $\boldsymbol{a}_{ik}$. For example, the position of an object's center of gravity is given by the mean point of its mass, and an axis pointing down in the direction of gravity. For an articulated object, such as a hand winch or door handle, the position and axis of rotation of the revolute joint defines another center. Although an object may have many centers, usually only one center is used for predicting an interaction. In this paper, we only consider a single object center $\boldsymbol{c}_i$, and leave automatically selecting the relevant center to future work.

Once the contact points have been found, they need to be defined with respect to the center's coordinate frame. If the axes of the center already defines three orthogonal axes $\boldsymbol{a}_i^x$, $\boldsymbol{a}_i^y$, and $\boldsymbol{a}_i^z$ this step is trivial. However, the center of gravity or the center of a revolute joint only define a single axis $\boldsymbol{a}_i^x$ and not a full 3D coordinate frame. In order to define the other two axes, we first project the contact points into a 2D plane, with the normal of the plane given by the first axis of the center $\boldsymbol{a}_i^x$. We then compute the matrix of second moments about the center position for the contact points, and subsequently compute the eigenvectors of the matrix. The second axis $\boldsymbol{a}_i^y$ is defined by the eigenvector with the largest eigenvalue, such that the mean of the contact points is in the positive direction. Using this approach, the contact point clouds are aligned according to the radial direction with the largest variance. The third axis is simply given by the cross product of the first two $\boldsymbol{a}_i^z = \boldsymbol{a}_i^x \times \boldsymbol{a}_i^y$.

The positions of the $\tilde{n}_i$ contact points in the object center's coordinate frame are denoted as $\tilde{\boldsymbol{p}}_{ij}$ with corresponding normals $\tilde{\boldsymbol{u}}_{ij}$ for $j \in \{1, \ldots, \tilde{n}_i\}$.

4.2.4 Computing Contact Distributions

Having computed a set of contact points, we now want to compare this set of contacts to previously observed ones. Rather than comparing points individually, we first model the set of

contact points as a distribution. In particular, we model them as a 6D Gaussian distribution, where the first three dimensions correspond to the positions of points, and the last three model the normals. In the lifting experiment in Section 4.3, we also investigate replacing the normals of each point with an estimate of the force. However, the forces are in most cases not known, especially when the interaction is between two objects and not with the robot.

Given a set of contacts, we now define a distribution over contact points as a Gaussian distribution. The mean vector $\boldsymbol{\mu}_i$ and variance $\boldsymbol{\Sigma}_i$ of the distribution are given as

$$\boldsymbol{\mu}_i = \frac{1}{\tilde{n}_i} \sum_{k=1}^{\tilde{n}_i} \begin{bmatrix} \tilde{\boldsymbol{p}}_{ik} \\ \tilde{\boldsymbol{u}}_{ik} \end{bmatrix},$$

$$\boldsymbol{\Sigma}_i = \frac{1}{\tilde{n}_i} \sum_{k=1}^{\tilde{n}_i} \left(\begin{bmatrix} \tilde{\boldsymbol{p}}_{ik} \\ \tilde{\boldsymbol{u}}_{ik} \end{bmatrix} - \boldsymbol{\mu}_i \right) \left(\begin{bmatrix} \tilde{\boldsymbol{p}}_{ik} \\ \tilde{\boldsymbol{u}}_{ik} \end{bmatrix} - \boldsymbol{\mu}_i \right)^T.$$

This model provides a compact representation of the mean contact position and normal orientation, as well as the correlations between the parameters around this mean.

4.2.5 Kernel Between Contact Distributions

Having converted the contact points into a contact distribution, we can now use a kernel to compute the similarity between distributions. We use the Bhattacharyya kernel [34] which is given by

$$k((\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i), (\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)) = \int \sqrt{\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)} \sqrt{\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \mathrm{d}\boldsymbol{x}.$$

The computation of the kernel is given in [35], and we include it again here for completeness. The kernel function is computed as

$$k((\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i), (\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)) = C \exp\left(-M/4\right),$$

where the values of C and M are given by

$$C = 0.5^{-d/2} |\hat{\boldsymbol{\Sigma}}|^{1/2} |\boldsymbol{\Sigma}_i|^{-1/2} |\boldsymbol{\Sigma}_j|^{-1/2},$$

$$M = \boldsymbol{\mu}_i^T \boldsymbol{\Sigma}_i^{-1} \boldsymbol{\mu}_i + \boldsymbol{\mu}_j^T \boldsymbol{\Sigma}_j^{-1} \boldsymbol{\mu}_j - \hat{\boldsymbol{\mu}}^T \hat{\boldsymbol{\Sigma}} \hat{\boldsymbol{\mu}}.$$

The vector $\hat{\boldsymbol{\mu}}$ is given by $\hat{\boldsymbol{\mu}} = \boldsymbol{\Sigma}_i^{-1} \boldsymbol{\mu}_i + \boldsymbol{\Sigma}_j^{-1} \boldsymbol{\mu}_j$, and the matrix $\hat{\boldsymbol{\Sigma}}$ is computed as $\hat{\boldsymbol{\Sigma}} = (\boldsymbol{\Sigma}_i^{-1} + \boldsymbol{\Sigma}_j^{-1})^{-1}$. The parameter $d = 6$ is the dimensionality of the Gaussians. The kernel function computes a value from zero to one, where a value of one is achieved if the contact distributions are identical. As the overlap between the distributions decreases, the kernel function tends to zero.

4.2.6 Extension to Multiple Gaussians

Although we focus on representing contact distributions using single Gaussians, the proposed framework is straightforward to extend to multiple Gaussians. By representing the contact distribution as a mixture of Gaussians, the model can capture more details of the distribution.

The resulting kernel can therefore distinguish between different contact distributions more easily.

However, the Bhattacharyya kernel is not suitable for comparing Gaussian mixture models. Instead, given that the contact distribution of object O_i has the form

$$f_i(\boldsymbol{x}) = \sum_{h=1}^{H_i} \nu_{ih} \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_{ih}, \boldsymbol{\Sigma}_{ih}),$$

where ν_i are the mixture components of the H_i Gaussians, one can compute the kernel function

$$k(f_i(\boldsymbol{x}), f_j(\boldsymbol{x})) = \frac{\int f_i(\boldsymbol{x}) f_j(\boldsymbol{x}) \mathrm{d}x}{\sqrt{\int f_i(\boldsymbol{x}) f_i(\boldsymbol{x}) \mathrm{d}x} \sqrt{\int f_j(\boldsymbol{x}) f_j(\boldsymbol{x}) \mathrm{d}x}},$$

in closed-form. This kernel function also has a value of 1 when the contact distributions are the same, and tends to zero as the overlap decreases. The kernel is based on the expected likelihood kernel [35] and is closely related to the Cauchy-Schwarz divergence [36].

4.2.7 Interaction-Specific Contact Similarity

Although the contact distribution is defined in a 6D space, not all of the dimensions will be equally relevant for predicting a given interaction. For example, when pushing open a door, the horizontal distance from the axis of rotation is more relevant than the vertical position along the axis. As a result, two contacts are more similar if they are offset vertically rather than horizontally from each other.

We can model this additional similarity by adding interaction-specific Gaussian noise $\mathcal{N}(\mathbf{0}, \tilde{\boldsymbol{\Sigma}})$ to the contact points. Thus, each contact point is represented as a Gaussian distribution $\mathcal{N}([\ \tilde{\boldsymbol{p}}_{ik}^T \quad \tilde{\boldsymbol{u}}_{ik}^T\ ]^T, \tilde{\boldsymbol{\Sigma}})$ instead of just a single point. If the offset between two contact points corresponds to a direction with a larger variance, then their distributions will overlap more and they will be considered as more similar. In practice, the interaction-specific covariance matrix $\tilde{\boldsymbol{\Sigma}}$ is added to the standard covariance matrices $\boldsymbol{\Sigma}_i$ and $\boldsymbol{\Sigma}_j$ before computing the kernel value. The experiment in Section 4.3.2 shows that the robot can use this additional similarity information to increase the sample efficiency of the learning algorithm.

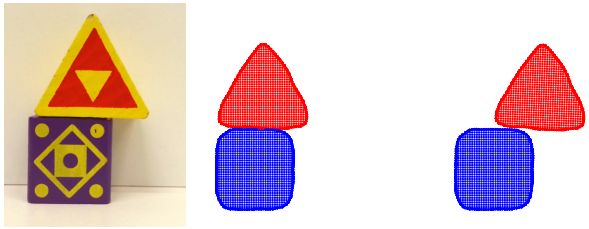
4.2.8 Classifying Contact Distributions

Having defined a kernel between contact distributions, we can now use a wide range of kernel methods from machine learning [73]. In order to classify a contact distribution, we use kernel logistic regression. Kernel logistic regression uses the similarity to previously observed distributions, with known labels, to classify new contact distribution. The probability that a contact distribution $\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ allows for a certain interaction $\mathcal{I}$ is given by

$$p(\mathcal{I}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) = \left(1 + \exp\left(\alpha\right)\right)^{-1},$$

where

$$\alpha = \theta_0 + \sum_{j=1}^{m} \theta_j k((\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i), (\boldsymbol{\mu}'_j, \boldsymbol{\Sigma}'_j)),$$



Positive Example Negative Example

Figure 4.5: Point cloud examples of a stable and an unstable stacking of blocks

and we have m previous examples of contact distributions $\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}'_j, \boldsymbol{\Sigma}'_j)$. The weight parameters θ can be learned using iterative reweighted least squares. Contact distributions that are not similar to any previous distributions will have a probability defined by θ_0. As kernel logistic regression is a probabilistic classifier, it can model a contact distribution that only sometimes allows for the interaction. Previous contact distributions that allowed for the interaction will generally have more negative weights, which will result in a probability closer to one.

4.3 Experiments

The proposed approach was implemented on a real robot, as shown in Fig. 4.1. The robot consists of two Kuka lightweight robot arms, each equipped with a DLR five-fingered hand [12], and a kinect. The robot was evaluated on two tasks: picking up an elongated object, and stacking assorted toy blocks.

4.3.1 Picking up Elongated Objects

In the first experiment, we applied the framework to the problem of predicting whether a given grasp allows an elongated object to be steadily lifted.

Experimental Setup

The robot performed 60 randomly selected grasps along the length of a spaghetti box. The first half of the grasps were performed with a three-fingered grasp and the other 30 were executed with a four-fingered grasp, as shown in Fig. 4.2. The robot subsequently tried to lift the box 13 cm above the table. The picking up of the box was considered successful if the object was no longer in contact with the table, and a failure otherwise, as shown in Fig. 4.3. Before lifting the box, the robot recorded the state of the scene and computed the contact distribution. Based on this information, the robot had to predict whether or not the lift would be successful. In order to detect contact points, we labeled ten points in one scene to train the contact classifier. The contact distribution is defined relative to the center of gravity.

In addition to evaluating the method explained in Section 4.2, referred to here as NORMAL+POS, we also evaluated several benchmark approaches. The first benchmark approach, MEANONLY, performs the classification using only the mean contact $\boldsymbol{\mu}_i$. The POS approach uses only the position distribution of the contact points and not the normals. As a result, the

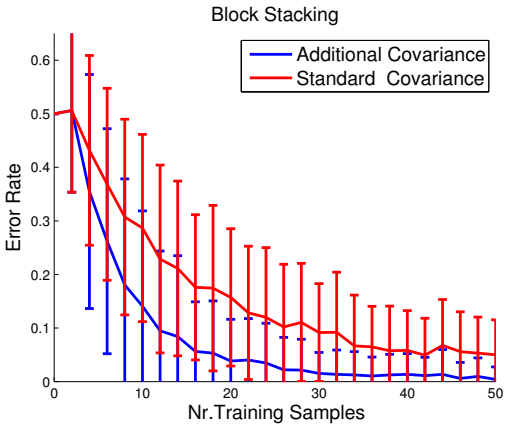


Figure 4.6: The expected error rate for the block stacking task. The red line indicates the performance when using the standard covariance matrix. The blue line shows the performance when adding the interaction-specific covariance matrix. The error bars indicate one standard deviation.

contact distribution is only 3D. Although the fingers do not have tactile sensors, forces can be roughly approximated using the joint torque sensors of the fingers and the relative positions of the contact points. The FORCE+POS approach is the same as NORMAL+POS, except that the normals $\boldsymbol{u}_i$ have been replaced by force estimates. The final method HANDRELATIVE uses the positions and estimated forces of the contact points, but defines the contact distribution relative to the hand rather than the object center.

The performance of the various methods were tested for different numbers for training samples. In each evaluation, ten grasps were selected as test samples. From the remaining grasp samples, a subset of samples were selected as training data. The classifier was then trained on the training data and used to classify the test samples. The error rate is given by the percentage of correctly classified grasps in the test set. This process was repeated 250 times for each classifier and each number of training samples. The results of the evaluation are shown in Fig. 4.4.

Discussion

Using only the mean contact or the distribution relative to the hand resulted in poor performance. The task was especially challenging for the HANDRELATIVE approach, as the object has the same shape along its length. Despite this challenge, the approach still obtained an error rate of 25.04%.

Using only the position of the contact points relative to the object center resulted in an error rate of 18.36%, which is only marginally better than the performance of HANDRELATIVE. In comparison, the NORMAL+POS and the FORCE+POS achieved error rates of 4.88% and 5.28% respectively. The contact normals clearly capture a considerable amount of information, as they allow side contacts to be differentiated from top contacts.

Both NORMAL+POS and FORCE+POS performed well on the task, and learned to accurately predict steady lifts. However, both approaches also have their limitations. The NORMAL+POS approach cannot differentiate between the robot gently placing its fingers on

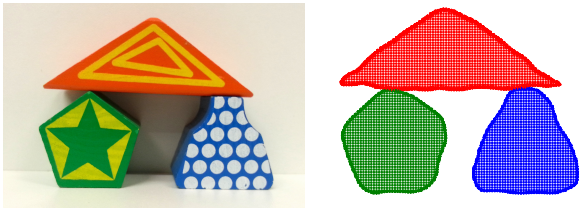


Figure 4.7: An example scene with three objects, wherein the green and blue objects are supporting the triangular red block.

the box and the fingers applying forces at the contacts. This approach can therefore sometimes only predict whether an interaction is possible, given the contacts, but not if the interaction is being performed. The FORCE+POS approach can differentiate between these two scenarios, and using it together with tactile sensing is a promising direction for future research. However, as the forces between objects will often not be directly observed, the NORMAL+POS approach is generally more applicable.

4.3.2 Stacking Objects

In the second experiment, the robot was given the task of classifying whether one object was supporting another. The robot then used the trained classifier to stack assorted toy blocks.

Classifying Stable Block Placements

The robot was provided with 60 example scenes, each containing two interacting toy blocks, such as the ones shown in Fig. 4.5. For the 30 negative examples, physically impossible static scenes were created by hand. The models of the blocks were acquired using a turn table setup and a kinect. The object center is again defined by the center of gravity. To train the contact point classifier, ten points were hand labelled in one scene. The points of the object were classified as contacts based on the features described in Section 4.2.2. Using additional features, such as the position and orientation of the points relative to the object's center, were also tested, but had no significant effects on the outcome of the experiment.

The performance of the contact point classifier was evaluated in the same manner as for the previous experiments. A set of ten test samples were randomly selected and removed from the pool of 60 samples. A subset of the remaining samples were then used to train the classifier. The classifier was subsequently applied to the ten test samples, and the error rate was recorded. The error rate is 1 if all ten samples were incorrectly classified, and 0 if all of them were correctly classified. The test samples were subsequently put back into the pool of samples. This process was repeated 250 times for each number of training samples.

In addition to the standard approach, we also evaluated adding an interaction-specific covariance matrix $\tilde{\Sigma}$, as explained in Section 4.2.7. The elements of the diagonal matrix were recomputed for each trial using a basic hill-climbing approach to minimize the leave-one-out cross-validation error rate on the training set.

The results of this experiment are shown in Fig. 4.6. Starting with error rates close to 50%, the classifiers' performances gradually improves as more samples are provided. Given 50 samples, the standard classifier achieved an expected error rate of 5.0%, and could accurately

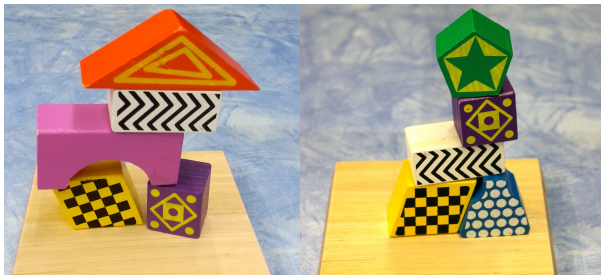


Figure 4.8: Two examples of block towers constructed by the robot.

predict when the object was being supported. Using the additional interaction-specific covariance matrix, the classifier achieved an expected error rate of 0.4% for 50 samples, and only required 20 samples to achieve an expected error rate of 3.84%. The sample efficiency of the algorithm can therefore be increased by incorporating the interaction-specific covariance. In many of the trials, the covariance matrix $\tilde{\Sigma}$ indicated that the vertical position of the supporting contacts was less relevant than the horizontal position. The experiment demonstrates the classifier's ability to generalize between different object shapes.

Generalization to Multiple Objects

In order to demonstrate the classifier's ability to generalize to multiple objects, it was applied to the scene of three objects shown in Fig. 4.7. In this scene, the top object is being supported by both of the lower objects. When the classifier is applied to the top block and only one of the bottom blocks, the interaction is classified as not supporting. However, we can also combine the blue and green point clouds of the bottom objects in order to create one compound object. When applying the classifier to the top object and this compound object, the top object is labeled as being supported by the bottom object. Thus, as one would expect, the classifier detects that the top is being supported by both objects jointly, and by neither one separately. The classifier was tested on two more similar scenes of three blocks, with the same results.

Building Block Towers

In the final part of the experiment, the real robot used the classifier from the first part to perform block stacking. The interaction-specific covariance matrix was not used in this experiment. The robot was provided with a small wooden board, on which to stack the blocks. In order to avoid all of the blocks being placed directly on the board, the placing of the blocks was limited to a single strip along the middle of the board. For every block, the robot observed the current scene using the kinect and used the resulting point cloud as the supporting object in the interaction. As the focus is not on the planning aspects of the problem, the sequence of blocks was predefined.

In order to determine a suitable placement for the current block, the robot sampled different positions in the scene. For each sample, the contact points were estimated and the probability of the block being supported was computed. The robot then attempted to place the block at the position with the highest probability.

Randomly sampling positions in the scene led to poor performance. One of the main

challenges for the robot was the noisy partial point cloud of the current scene. The kinect usually only captured the top and front of the current block stack, but not the back or sides. The lack of reliable points on the sides of objects resulted in unforeseen collisions between blocks. This problem could be alleviated by obtaining more views of the scene, completing the point cloud based on symmetries [5, 47], or applying a penalty for placing the block into occluded regions.

In order to reduce the number of accidental collisions, we also implemented a sampling approach that mimics the movement of the block when it is being put down. The robot sampled 20 horizontal positions at 7.5mm increments across the width of the board. For each horizontal position, the robot sampled vertical placements at 5mm increments in a top-down manner until contact was detected between the block and the stack.

In order to evaluate the proposed approach, the robot was given the task of creating five towers consisting of five blocks each. Using the improved sampling approach, the robot successfully placed 96% of the blocks without knocking any blocks down. Only one block was misplaced by a few millimeters and fell down. The robustness of the system could be further improved by also considering the probability of success of neighboring positions [7].

The robot currently ignores the interactions between blocks further down in the stack. As a result the robot may select a block placement that causes a supporting block to fall down. One potential solution to this problem would be to recheck the interactions between objects further down the stack. For each interaction, the objects higher up in the stack would then be treated as a single compound object, with a corresponding object center. This approach would however require the robot to keep a model of the current scene's geometry.

The results of the experiment show that the robot was able to construct multiple block towers, such as the ones shown in Fig. 4.8,using the proposed approach. A video of the robot stacking blocks is avaliable at: http://youtu.be/6S5eJgE28sg

4.4 Conclusions

In this paper, we presented a kernel-based approach to learning object interactions from contact distributions. The proposed approach is based on modeling the distribution of contact points as a Gaussian distribution. The Bhattacharyya kernel is then used to compute the similarity between the contact distributions. In the experiments, we used kernel logistic regression to predict stable grasps of objects, as well as suitable placements of objects. Using the learned classifier, the robot was able to build small towers out of assorted blocks.

Chapter 5

Learning Inverse Dynamics Models with Contacts (TUD)

Abstract

In whole-body control, joint torques and external forces need to be estimated accurately. In principle, this can be done through pervasive joint-torque sensing and accurate system identification. However, these sensors are expensive and may not be integrated in all links. Moreover, the exact position of the contact must be known for a precise estimation. If contacts occur on the whole body, tactile sensors can estimate the contact location, but this requires a kinematic spatial calibration, which is prone to errors. Accumulating errors may have dramatic effects on the system identification. As an alternative to classical model-based approaches we propose a data-driven mixture-of-experts learning approach using Gaussian processes. This model predicts joint torques directly from raw data of tactile and force/torque sensors. We compare our approach to an analytical model-based approach on real world data recorded from the humanoid *iCub*. We show that the learned model accurately predicts the joint torques resulting from contact forces, is robust to changes in the environment and outperforms existing dynamic models that use of force/torque sensor data.

5.1 Introduction

A key challenge for torque-controlled humanoid robots is to accurately estimate their dynamics in presence of contacts, e.g., during manipulation in clutter [32], whole-body movements [38] or ground contacts in locomotion [9]. Analytic dynamics models suffer from inaccurate parameter estimation, unmodeled dynamics (e.g., friction, couplings, elasticities) and noisy sensor measurements. With contacts the problem is even more challenging due to discontinuities and additional non-linearities, which are difficult to model or estimate. Moreover, if contact locations are not fixed a priori or known with sufficient precision, small errors in the localization of the external force can substantially deteriorate the inverse dynamics computation [19].

Nevertheless, many modern control strategies like inverse dynamics control [22], computed torque control [74] or model predictive control [56] rely on accurate dynamic models. With inaccurate dynamics models they can produce suboptimal policies by not taking external forces

into account, which are caused by contacts.

As a first step toward a more informed controller that explicitly considers the effect of contacts, we propose to learn the inverse dynamics model from tactile sensor readings and force-torque sensors. In contrast to classical techniques based on the identification of dynamics parameters [82, 59, 77], we propose a fully data-driven machine learning approach based on non-parametric models, where both the rigid body dynamics as well as the effect of external forces on the robot structure are learned directly from data collected on the real robot. The proposed model makes use of the raw sensor data and does not require a kinematic/dynamics calibration [82, 59, 77]. In particular, it does not need a spatially calibrated model of the skin [18]. We propose to use a mixtures-of-experts based on Gaussian Processes (GP) to learn the non-linear system dynamics. Each of these GP experts models a single contact "type" and can be learned straightforwardly. By using a gating network that activates and deactivates the individual GP experts we can switch between contact models and generalize to more complex environments. We evaluate our model learning approach on the arm of the *iCub* humanoid robot [55] (see Fig. 5.1) and compare to a state-of-the-art model-based approach. The learned inverse dynamics model outperforms the analytic approach and we demonstrate that the learned model can generalize to changing contact locations. To the best of our knowledge this is the first demonstration of how joint torques can be learned on a humanoid robot equipped with tactile and force/torque sensors in presence of contacts.

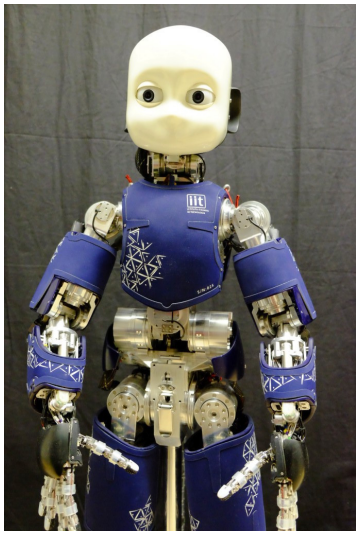


Figure 5.1: The humanoid robot *iCub* used in the experiments.

5.2 Problem Formulation

The inverse dynamics of a robot with m degrees of freedom can be generally described by

$$\boldsymbol{\tau} = \underbrace{\boldsymbol{M}(\boldsymbol{q})\,\ddot{\boldsymbol{q}} + \boldsymbol{h}(\boldsymbol{q},\dot{\boldsymbol{q}})}_{\boldsymbol{\tau}_{\text{RBD}}} + \epsilon(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}})\,, \tag{5.1}$$

where $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$ and $\ddot{\boldsymbol{q}}$ are the joint positions, velocities and accelerations, respectively, $\boldsymbol{M}(\boldsymbol{q})$ is the inertia matrix and

$$\boldsymbol{h}(\boldsymbol{q},\dot{\boldsymbol{q}}) = C(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + g(\boldsymbol{q}) + F_v\dot{\boldsymbol{q}} + F_s\,\text{sgn}(\dot{\boldsymbol{q}})$$

is the matrix combining the contributions from Coriolis and centripetal, friction (viscous and static) and gravity forces. The term $\epsilon(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}})$ in (5.1) captures the errors of the model, such as unmodeled dynamics (e.g., elasticities and Stribeck friction), inaccuracies in the dynamic parameters (e.g., masses, inertia), vibrations, couplings, and sensor noise. With a set $\mathcal{C} = \{c_1 \dots c_n\}$ of contacts c_i between the robot and the environment, (5.1) becomes

$$\boldsymbol{\tau} = \underbrace{\boldsymbol{M}(\boldsymbol{q})\,\ddot{\boldsymbol{q}} + \boldsymbol{h}(\boldsymbol{q},\dot{\boldsymbol{q}})}_{\boldsymbol{\tau}_{\text{RBD}}} + \epsilon(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}}) + \sum_{c_i\in\mathcal{C}} \boldsymbol{J}_{c_i}^{\top}(\boldsymbol{q})\,\boldsymbol{\gamma}_i\,, \tag{5.2}$$

where the last term accounts for the additive effect of the external wrenches (forces and moments) $\boldsymbol{\gamma}_i$ applied at contact location c_i, and $\boldsymbol{J}_{c_i}(\boldsymbol{q})$ is the contact Jacobian. Note that the contact location c_i is not necessarily fixed as the contacts may occur on the whole robotic structure and not exclusively at the end-effectors. In such a case, the contact location, if not known a priori, must be estimated, typically through distributed tactile sensors. To compute the contact Jacobian, we need the position of the contact point with respect to the reference frame of the link [26]. Such a knowledge requires a kinematic calibration of the skin as explained in [18].

5.2.1 Classical model-based approaches for computing the robot dynamics

Classical approaches for computing $\boldsymbol{\tau}$ or $\boldsymbol{\tau}_{\text{RBD}}$ rely on the dynamics model with known or identified kinematics and dynamics parameters [31]. The torques $\boldsymbol{\tau}_{\text{RBD}} = \boldsymbol{M}(\boldsymbol{q})\,\ddot{\boldsymbol{q}} + \boldsymbol{h}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ can be computed analytically through the rigid body dynamics model of the robot, a standard parametric description of the robot [24]. The term $\epsilon(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$ is often neglected, or implicitly taken into account by considering a perturbation in the dynamics parameters of $\boldsymbol{\tau}_{\text{RBD}}$, which need to be identified accurately.

Although parameter identification for industrial robots is relatively easy with exciting trajectories [64], the procedure for floating-base robots, such as humanoids, is not straightforward because of two main issues: 1) The generation of sufficiently large accelerations for the identification while maintaining the robot balance and the control of contacts. This issue was well explained by Yamane [82], who proposed a technique to identify the mass and the local COM of the links in a humanoid robot with fixed feet at the ground and slow joint trajectories. 2) The measurement of the external forces $\boldsymbol{\gamma}_i$ exerted on the robot. Note that it may not be straightforward to measure the external forces $\boldsymbol{\gamma}_i$ as it is not possible to cover the robot body with 6-axis force/torque sensors to measure the force exerted on every possible contact location c_i. Usually, such sensors are big, heavy and expensive. Thus, they are carefully placed where the external forces are critical for the main tasks. In such a case, it is possible to identify the dynamics parameters while balancing and walking without additional contacts [59]. When force/torque sensors are placed proximally, such as in the *iCub* arms [26], some of the dynamics parameters can be identified, but in absence of contacts [77].

When multiple contacts are exerted on the robot structure at locations other than the classical end-effectors, it is still possible to compute a precise inverse dynamics model, but this requires both pervasive joint torque sensing, such as in *Toro* [59], and additional force/torque and tactile sensing, such as in *iCub* [30]. Moreover, it requires the precise knowledge of the contact locations detected by the tactile sensors, which necessitates a spatial calibration of the skin [18]. This procedure is prone to errors, and it has been shown that small errors in the kinematics calibration of the taxels (i.e., the tactile units) can induce non-negligible errors in the estimation of the contact forces [19].

Generally, these model-based approaches have three main limitations: 1) It is hard to add details about couplings, elasticity, friction and other nonlinear dynamics, which are required for high accuracy; 2) The performance of the data-driven identification strongly depends on the experimental setting (with/without contacts) and the exciting trajectories [64]; 3) They make strong assumptions to handle contacts.

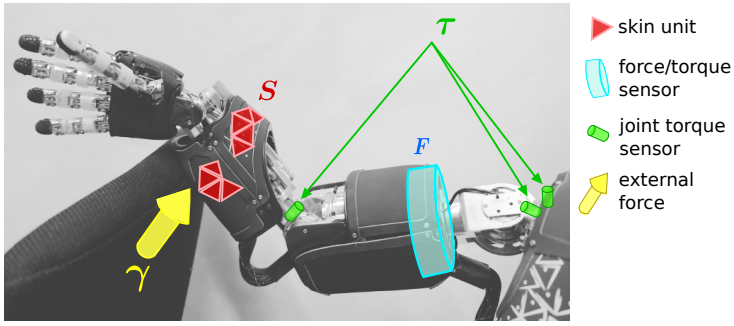


Figure 5.2: Illustration of the force/torque and tactile sensors during a contact of the robot arm with the environment.

5.2.2 Learning the inverse dynamics

An alternative and appealing approach to analytic dynamics computation is to use machine learning methods to learn the dynamics model of a robot [58, 80, 16]. Without the need for compensating for inaccurate dynamics parameters and accumulated errors, a learned dynamics model can improve the tracking and control performances of a robot, as shown in [57] for an industrial manipulator. The clear advantage of learning the inverse dynamics is that we can overcome the limitations of the aforementioned approaches: difficulty in modeling complex nonlinear dynamics, impossibility to generate suitable exciting trajectories, restrictive assumptions regarding contacts and sensors, prior accurate kinematics calibration of the tactile sensors. Despite the success of learned dynamics models in robotics, to the best of our knowledge there are no examples in the literature where dynamic contacts are also learned. The inclusion of dynamic contact models in the dynamics highlights two main problems: First, switching from a no-contact model to a contact-model requires to observe the system state and to model a discontinuous function [76]. Second, switching between different contacts $c_i \in \mathcal{C}$ must be properly handled.

Here, we provide a first formulation to this problem, and we show that it is possible to learn the inverse dynamics model of the arm of the *iCub* robot by means of proximal force/torque measurements $\boldsymbol{F}$ and distributed tactile sensors $\boldsymbol{S}$ (without requiring a spatially calibrated model of the skin [18]).

5.3 Learning Inverse Dynamics with Contacts

In this section, we present our proposed approach to learning inverse dynamics with contacts. We first formalize the problem as learning a mixture-of-experts model. Then we detail how we implement Gaussian processes as the corresponding experts.

5.3.1 Learning contacts as a mixture-of-experts

When learning inverse dynamics with contacts (5.2), we assume that the (contact-free) inverse dynamics from (5.1) can be computed precisely, either from an analytical model or from a learned model [57]. In our experiments, we employ a learned GP model as contact-free inverse

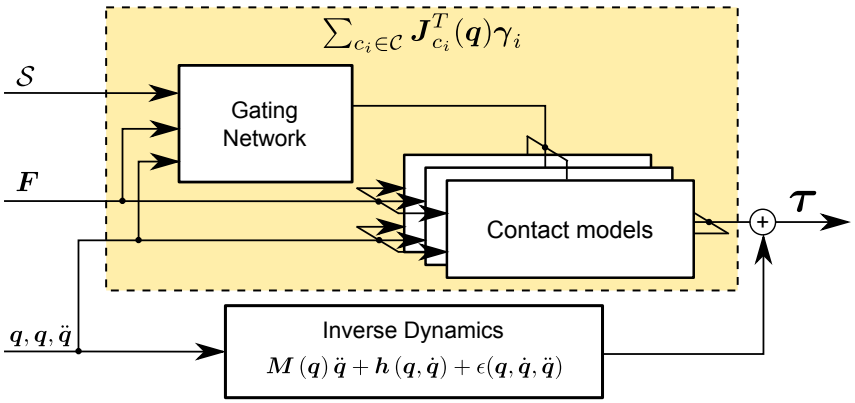


Figure 5.3: Our approach extends existing inverse dynamics without contacts by learning many contact models, which serve as correction terms under different contact types. The decision of which contact model to activate is made by a gating network, which uses skin measurements $\boldsymbol{S}$, the force torque sensors $\boldsymbol{F}$ and the current state $\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}$.

dynamics. The reason for this choice are the unmodeled dynamics $\epsilon(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$, which introduce substantial errors even without contacts. As a result of the pre-existing contact-free inverse dynamics, only the model of the residual term of the external forces $\sum_{c_i \in \mathcal{C}} \boldsymbol{J}_{c_i}^{\top}(\boldsymbol{q})\,\boldsymbol{\gamma}_i$ has to be separately learned. In this paper, we consider a robot that is provided with skin measurements $\boldsymbol{S}$ from the tactile sensors, force measurements $\boldsymbol{F}$ from the force torque sensors (FTS) and the ground truth of the torques $\boldsymbol{\tau}$ from the joint torque sensors (JTS). An illustration of these relevant components is shown in Fig. 5.2. Modeling the external forces $\sum_{c_i \in \mathcal{C}} \boldsymbol{J}_{c_i}^{\top}(\boldsymbol{q})\,\boldsymbol{\gamma}_i$ can be formalized as the regression task

$$\boldsymbol{y} = f([\boldsymbol{q}, \boldsymbol{S}]) + w\,, \tag{5.3}$$

where $\boldsymbol{y} = \sum_{c_i \in \mathcal{C}} \boldsymbol{J}_{c_i}^{\top}(\boldsymbol{q})\,\boldsymbol{\gamma}_i$ and w is an i.i.d. Gaussian measurement noise with mean 0 and variance σ_w^2. Contacts with different parts of the body lead to different effects in the dynamics. Intuitively, it is necessary to consider the skin input $\boldsymbol{S}$ to identify the position of the contact. Additionally, measurements of the force applied by the contacts are necessary to deal with a non-static environment. Theoretically, these measurements can be provided by the skin. However, the artificial skin used in our experiments does not provide a precise six-dimensional measure of the contact force. Therefore, in the implementation of our model we substitute the force measurement from the skin with the force/torque measurements $\boldsymbol{F}$. The corresponding regression problem (5.3) is complicated due to the high-dimensional space of the input $\boldsymbol{x} \in \mathcal{X}$ (the skin measurements $\boldsymbol{S}$ alone account for hundreds of dimensions). Therefore, we rephrase this regression task as a problem of learning a mixture-of-experts model. With this model, we decompose (5.3) as

$$\sum\nolimits_{c_i \in \mathcal{C}} \boldsymbol{J}_{c_i}^{\top}(\boldsymbol{q})\,\boldsymbol{\gamma}_i \;=\; \sum\nolimits_{j \in \mathcal{J}} f_j([\boldsymbol{q}, \boldsymbol{F}]) + w\,, \tag{5.4}$$

where $\mathcal{J}$ is the set of active experts f_j. Note that the skin input $\boldsymbol{S}$ is no longer explicitly part of the inputs of the experts. Therefore, each single expert f_j is now sufficiently low-dimensional to be modeled independently. At the same time the possibility of summing the contribution

of each contact allows to account for complex behaviors. As single expert f_j we use Gaussian processes for the mapping $[\boldsymbol{q}, \boldsymbol{F}] \mapsto \boldsymbol{J}_j^\top(\boldsymbol{q})\gamma_j$. A gating network is used to select the experts that are currently active and to add their contributions. An illustration of our approach is shown in Fig. 5.3. In this paper, we implement this gating network as a multi-class classifier $\mathcal{J} = g(\boldsymbol{q}, \boldsymbol{S}, \boldsymbol{F})$ that selects which contact is currently ongoing. For simple tasks, this gating network can be designed using heuristics (e.g., using thresholds on the activation of the tactile sensors). However, for more complex systems an adaptive, data-driven approach may be more suitable. In the experimental section we evaluate the learning of such gating network.

5.3.2 Gaussian processes as expert models

Gaussian Processes (GPs) [66] are a state-of-the-art regression method. They have been used in robotics to learn dynamics models [16] and for control [17]. In this paper, a GP is a distribution over inverse dynamics models $f \sim \mathcal{GP}\left(m_f, k_f\right)$, fully defined by a prior mean m_f and a covariance function k_f. We choose as prior mean $m_f \equiv \boldsymbol{\tau}_{\text{RBD}}$ and as covariance function k_f the squared exponential with automatic relevance determination and Gaussian noise

$$k(\boldsymbol{x}_p, \boldsymbol{x}_q) = \sigma_f^2 \exp\left(-\tfrac{1}{2}(\boldsymbol{x}_p - \boldsymbol{x}_q)^T \boldsymbol{\Lambda}^{-1}(\boldsymbol{x}_p - \boldsymbol{x}_q)\right) + \sigma_w^2 \delta_{pq}$$

where $\boldsymbol{\Lambda} = \text{diag}([l_1^2, ..., l_D^2])$ and δ_{pq} is the Kronecker delta (which is one if $p = q$ and zero otherwise). Here, l_i are the length-scales, σ_f^2 is the variance of the latent function $f(\cdot)$ and σ_w^2 the noise variance.

In our experiments, when learning contact models, the input is defined as $\boldsymbol{x} = [\boldsymbol{q}, \boldsymbol{F}]$, while the output (observations) $\boldsymbol{y} = \boldsymbol{\tau}$ are the torques. Hence, given n training inputs $\boldsymbol{X} = [\boldsymbol{x}_1, ..., \boldsymbol{x}_n]$ and corresponding training targets $\boldsymbol{y} = [y_1, ..., y_n]$, we define the training data set $\mathbb{D} = \{\boldsymbol{X}, \boldsymbol{y}\}$. Training the GP corresponds to finding good hyperparameters $\boldsymbol{\theta} = [l_i, \sigma_f, \sigma_w]$, which is done by the standard procedure of maximizing the marginal likelihood [66].

The GP yields the predictive distribution over torques for a new input $\boldsymbol{x}_* = [\boldsymbol{q}_*, \boldsymbol{F}_*]$

$$p(\boldsymbol{y}|\mathbb{D}, \boldsymbol{x}_*, \boldsymbol{\theta}) = \mathcal{N}\left(\mu(\boldsymbol{x}_*), \sigma^2(\boldsymbol{x}_*)\right) , \tag{5.5}$$

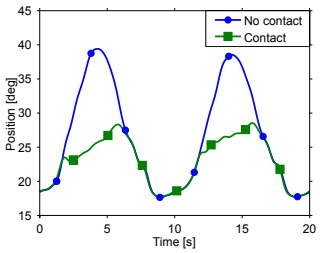
where the mean $\mu(\boldsymbol{x}_*)$ and the variance $\sigma^2(\boldsymbol{x}_*)$ are

$$\mu(\boldsymbol{x}_*) = \boldsymbol{k}_*^T \boldsymbol{K}^{-1} \boldsymbol{y} , \quad \sigma^2(\boldsymbol{x}_*) = k_{**} - \boldsymbol{k}_*^T \boldsymbol{K}^{-1} \boldsymbol{k}_* , \tag{5.6}$$

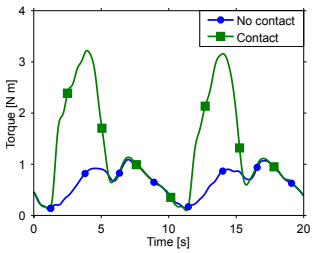
respectively. The entries of the matrix $\boldsymbol{K}$ are $K_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, and we define $k_{**} = k(\boldsymbol{x}, \boldsymbol{x})$ and $\boldsymbol{k}_* = k(\boldsymbol{X}, \boldsymbol{x})$.

5.4 Experimental Set-up and Evaluation

In this section, we describe the experimental setting and the humanoid robot *iCub* used in the experiments. We present four different experiments where we demonstrate that 1) Our approach can learn single contact models; 2) A single learned model (i.e., an expert) is robust to small changes in the position of the contact; 3) Our approach extends to multiple contacts by combining models of single contacts; 4) The gating network activating the experts can be learned to reduce the complexity of manually design it.



(a) Task space



(b) Torque

Figure 5.4: **Learning a single contact:** Effects of a contact (green curve) compared to the free movement without obstacle (blue curve). These effects are visible in the task space position (a) and in the torque measured by the joint torque sensor (b).

	Method	Shoulder 1 [Nm]	Shoulder 2 [Nm]	Elbow [Nm]
Full trajectory	*iDyn*	$0.09 \pm 1.1 \times 10^{-3}$	$0.16 \pm 1.8 \times 10^{-3}$	$0.05 \pm 7.4 \times 10^{-4}$
	Our model	$\mathbf{0.04 \pm 5.6 \times 10^{-4}}$	$\mathbf{0.07 \pm 9.8 \times 10^{-4}}$	$\mathbf{0.02 \pm 3.1 \times 10^{-4}}$
Contact only	*iDyn*	$0.07 \pm 3.1 \times 10^{-3}$	$0.13 \pm 5.7 \times 10^{-3}$	$0.08 \pm 3.0 \times 10^{-3}$
	Our model	$\mathbf{0.03 \pm 1.5 \times 10^{-3}}$	$0.12 \pm 5.9 \times 10^{-3}$	$\mathbf{0.03 \pm 1.3 \times 10^{-3}}$

Table 5.1: **Learning a single contact:** Mean and standard deviation of the mean for the RMSE of the test set for ground truth, predictions with the *iDyn* and our learned model. The learned model predicts the torque more accurately than *iDyn* for both the full trajectory and during the only contact phase.

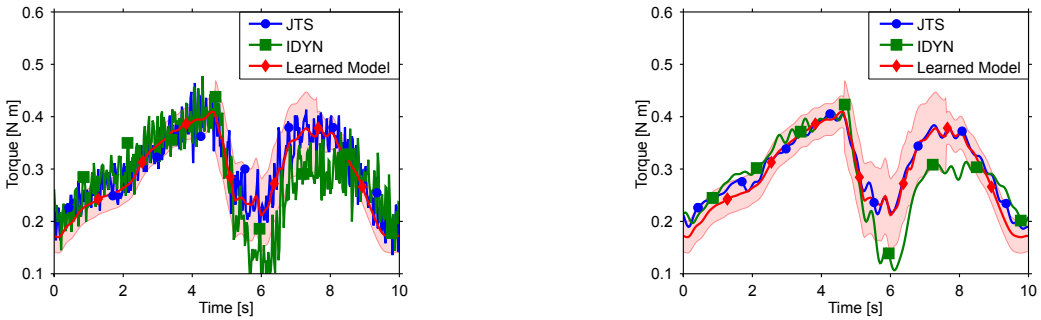
5.4.1 Experimental set-up

The experiments were conducted on the *iCub* [55], a humanoid robot with 53 degrees of freedom, sized as a child (104 cm tall, 24 kg of weight). This robot is equipped with several sensors: an inertial sensor in the head, four 6-axis force/torque sensors placed proximally in the middle of legs and arms, and an artificial skin consisting of many distributed tactile elements (taxels) mounted on the robot covers [10]. The information from these three types of sensors is used to estimate the joint torques and the external contact forces by the *iDyn* library [30]. In the following, $\boldsymbol{\tau}_{\mathrm{IDYN}}$ denotes the joint torques estimated by the *iDyn* library, which we use as analytical model for comparison. For more detail on its contact detection and taxels calibration we refer to [18, 19].

The *iCub* used in the experiments is equipped with three additional Joint Torque Sensors (JTSs), two in the shoulder and one in the elbow. The JTSs are calibrated by computing the offset and gain trough least-square regression with respect to the output of *iDyn*. We consider these calibrated JTSs as ground truth measurements of the joint torques $\boldsymbol{\tau}$. In our experiments, we used the *iCub* torso and arms (3 and 7 degrees of freedom, respectively) and the skin input $\boldsymbol{S}$ from the forearm, which consists of 270 sensor measurements.

5.4.2 Learning a single contact

In this experiment, we consider the *iCub* making contact with a single obstacle. The evaluation is performed on a simple tracking task with the *iCub*'s end-effector moving along a circular



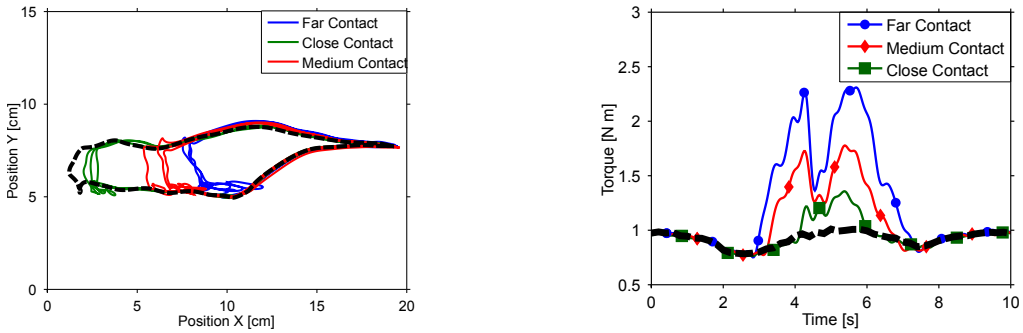
(a) Real data. (b) Filtered visualization.

Figure 5.5: **Learning a single contact**: Comparison of the torque measured at the elbow (with contact) by the JTS, estimated by *iDyn* and our learned model (shown as mean ± 2 std). Our learned model better predict the torque measured by JTS (a). Additionally, due to the identification of the noise in the model, its prediction is smoother compared to both the noisy JTS measurements and the prediction from *iDyn*. For visualization purposes we also show the predictions when filtering JTS and *iDyn* (b).

trajectory. We repeat the task twice: first without any contact and then with a contact at a fixed position. Fig. 5.4 shows the effects of the contact in terms of position and torque during the tracking task. When the contact occurs the position error increases considerably. As a result, the torque is increased to compensate for the obstacle. We collected 10 repetitions of the trajectory with the contact and used 8 of them to train the model. The remaining trajectories are used as test set to evaluate the predictive performances of our learned model. For this experiment we consider a single expert (the gating network still decides whether to activate the expert).

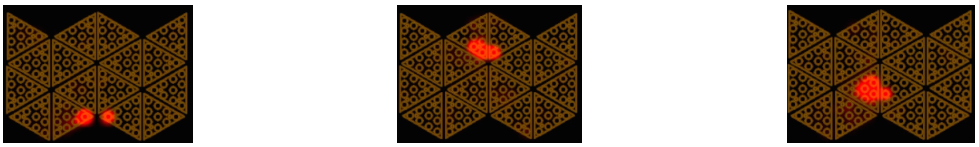
We compare the baseline joint torque (measured by the JTS) to the joint torque estimated by the analytic model *iDyn* and the joint torque $\boldsymbol{\tau}_{\text{IDM}}$ predicted by our learned model. In Table 5.1, we report the root mean square error (RMSE) and the standard deviation of the mean of *iDyn* and our learned model for all the three joints. Additionally, we report both the error of the learned models (learned RBD plus learned contact model) during the full trajectory and exclusively *during* the contact. In five out of six cases, the learned model performs better than the analytic model. In the sixth case (contact only, shoulder 2), the performance of the learned model is similar to the analytic model. However, increasing the amount of data used for training may further increase the performance of the learned model. A visual representation of the predictions of the test set for the elbow joints is shown in Fig. 5.5.

This experiment provides evidence that the classical rigid-body dynamics model $\boldsymbol{\tau}_{\text{RBD}}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$ and the *iDyn* estimation (that also exploits proximal force/torque sensing) fail to accurately estimate the joint torques when the robot is in contact with the environment. Moreover, we show that the learned contact model, when combined with the RBD model, provides a better approximation of the joint torque.



(a) Task space (b) Torque

Figure 5.6: **Robustness of the single contact model:** Effects of the contact on the task space and the torque for the three different contact types: contact 1 (far), contact 2 (medium) and contact 3 (close). The task in absence of contact is displayed as reference (**black dashed curve**).



(a) Contact 1 (b) Contact 2 (c) Contact 3

Figure 5.7: **Robustness of the single contact model**. The different contact locations detected by the forearm skin respectively for the three contacts: contact 1 (far), contact 2 (medium) and contact 3 (close).

5.4.3 Robustness of the single contact model

In the following, we show that the prediction performance of each GP expert is robust to small variations in the position of the contact. This is important since the exact position of the obstacle does not need to be known in advance (within a single expert f_j). As in the previous experiment we consider a tracking task along a circular trajectory. However, this time the obstacle is placed at one of three different positions along the trajectory: close, medium and far. Each of these obstacles is shifted 2 cm along the horizontal axis. Obstacles at different positions along the trajectory lead to different effects in terms of both joint position and torque signal, as clearly visible in Fig. 5.6. Note that the skin input $\boldsymbol{S}$ will also be affected, as shown in Fig. 5.7. Hence, we could potentially learn a separate expert for each contact. However, we only consider a single expert as we want to demonstrate the robustness of a single expert, not of the gating network.

The contact model is learned using the data collected from contact 1 and contact 3 (far and close contacts) and as validation the data set generated from the *unseen* contact 2 (medium) is used. In Table 5.2, the RMSE for all three contacts are reported for *iDyn* and our learned model, respectively. The results show that the learned model is robust to unseen contacts and performs equally well or better than the analytic model *iDyn*.

	Method	Shoulder 1 [Nm]	Shoulder 2 [Nm]	Elbow [Nm]
Far contact	*iDyn*	$0.13 \pm 3.9 \times 10^{-3}$	$0.40 \pm 9.7 \times 10^{-3}$	$0.06 \pm 1.9 \times 10^{-3}$
	Our model	$\mathbf{0.06 \pm 1.9 \times 10^{-3}}$	$\mathbf{0.08 \pm 2.9 \times 10^{-3}}$	$\mathbf{0.03 \pm 8.0 \times 10^{-4}}$
Close contact	*iDyn*	$0.09 \pm 2.2 \times 10^{-3}$	$0.22 \pm 4.5 \times 10^{-3}$	$0.04 \pm 0.9 \times 10^{-3}$
	Our model	$\mathbf{0.06 \pm 1.4 \times 10^{-3}}$	$\mathbf{0.06 \pm 1.4 \times 10^{-3}}$	$\mathbf{0.02 \pm 6.3 \times 10^{-4}}$
Medium contact	*iDyn*	$0.10 \pm 2.8 \times 10^{-3}$	$0.32 \pm 6.7 \times 10^{-3}$	$0.05 \pm 1.3 \times 10^{-3}$
	Our model	$\mathbf{0.06 \pm 1.7 \times 10^{-3}}$	$\mathbf{0.12 \pm 4.7 \times 10^{-3}}$	$0.05 \pm 1.7 \times 10^{-3}$

Table 5.2: **Robustness of the single contact model:** Errors between the ground truth (JTS) and the predictions with either the *iDyn* and our learned model on the test set. A single expert is robust to small variations of the contact.

	Method	Shoulder 1 [Nm]	Shoulder 2 [Nm]	Elbow [Nm]
Right contact	*iDyn*	$0.10 \pm 1.3 \times 10^{-3}$	$0.13 \pm 1.6 \times 10^{-3}$	$0.06 \pm 8.1 \times 10^{-4}$
	Our model	$\mathbf{0.04 \pm 6.3 \times 10^{-4}}$	$\mathbf{0.07 \pm 1.2 \times 10^{-3}}$	$\mathbf{0.02 \pm 2.7 \times 10^{-4}}$
Left contact	*iDyn*	$0.08 \pm 1.2 \times 10^{-3}$	$0.16 \pm 2.0 \times 10^{-3}$	$0.05 \pm 8.2 \times 10^{-4}$
	Our model	$\mathbf{0.03 \pm 5.7 \times 10^{-4}}$	$\mathbf{0.07 \pm 9.6 \times 10^{-4}}$	$\mathbf{0.02 \pm 2.8 \times 10^{-4}}$
Both contacts	*iDyn*	$0.10 \pm 1.3 \times 10^{-3}$	$0.11 \pm 1.4 \times 10^{-3}$	$0.07 \pm 8.4 \times 10^{-4}$
	Our model	$\mathbf{0.05 \pm 8.3 \times 10^{-4}}$	$\mathbf{0.10 \pm 1.6 \times 10^{-3}}$	$\mathbf{0.03 \pm 4.0 \times 10^{-4}}$

Table 5.3: **Learning multiple contacts:** Root mean square error between the ground truth (JTS) and the predictions with the *iDyn* and our learned model on the test set. Our learned model predicts the torque more accurately than *iDyn*.

5.4.4 Learning multiple contacts

After learning single contacts, we now show how to combine the learned models to adapt to unseen and more complex environments with multiple contacts. We consider a scenario having the *iCub* performing a circular motion with its left arm. We initially performed two experiments with an obstacle either on the left and on the right of the reference trajectory (see Fig. 5.8). With the data collected in these two contact cases, we trained two independent expert models f_1, f_2, one for each contact. We repeated the experiment, but this time with both left and right contacts and used this last unseen case to validate our models. Fig. 5.9 shows an example of the prediction and the corresponding activation of the two contact models. During both the right and the left contact, the corresponding experts are activated by the gating network. Therefore, we can successfully combine the contributions of the single contact models learned to generalize to unseen cases with multiple contacts. Table 5.3 reports the RMSE for the predictions. We notice that even in this experiment the experts accurately learn the effects of single contacts. Moreover, the gating network allows us to combine the experts to generalize to unseen environments, such as in the case of both contacts.

5.4.5 Learning the gating network

So far, we assumed a heuristic gating network to select the active experts. In this experiment we show that a learned gating network achieves a comparable accuracy as a manually devised heuristic. As ground truth to evaluate the performances, as well as for training the classifier,

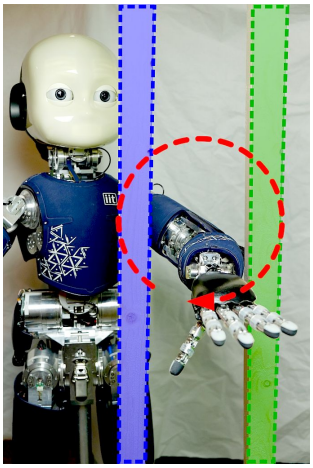


Figure 5.8: **Learning multiple contacts:** The robot performs a circle with its left arm. The forearm collides alternatively with the left, the right or both contacts.

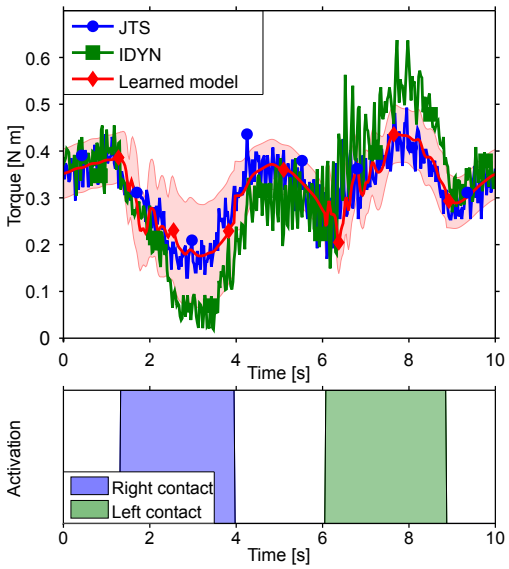


Figure 5.9: **Learning multiple contacts:** Prediction of torques with multiple contacts and the corresponding activation of the gating network. Our mixture-of-experts model combines single-contact models to a multiple-contact model.

we labeled the data with one of the following labels: no contact, left contact, right contact. The heuristic is based on thresholds of the activation of the skin input $\boldsymbol{S}$ and the force torque sensors $\boldsymbol{F}$. We train a Support Vector Machine (SVM) classifier (using the library LIBSVM [11]) having as input $\boldsymbol{q}, \boldsymbol{S}, \boldsymbol{F}$ and as output the contact labels (none, left, right).

We evaluated the performance of the trained classifier on an unseen test set. Fig. 5.10 shows that the learned SVM achieved a classification accuracy that is similar to the heuristic gating network. Equivalent results are obtained in terms of RMSE of the inverse dynamics when comparing the experts models learned by the gating networks. However, training the gating network (i.e., training the SVM classifier) requires considerably less expert knowledge compared to designing a heuristic. As there is no visible performance difference, we conclude that training the gating network is generally preferable. Increasing the number of training data may further increase the accuracy of the gating network.

5.5 Conclusions

Whole-body control strategies that exploit contacts need accurate models of the system dynamics. This is crucial for balance and stabilization, and to increase the number of potential actions that the robot is able to execute, e.g., creating a contact to reach for distant objects. We introduced a data-driven mixture-of-experts approach based on Gaussian processes for learning inverse dynamics models with contacts. We evaluated our model on the *iCub* humanoid robot using tactile sensors and force/torque sensors as model inputs. We showed that the model accurately predicts contact forces and outperforms a state-of-the-art analytical approach used to estimate the joint torques in *iCub*. The estimation from the learned model

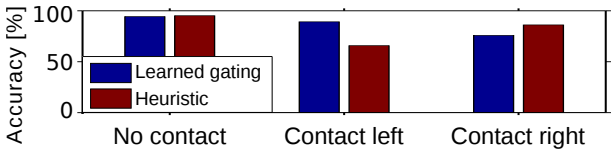


Figure 5.10: **Learning the gating network:** Classification accuracy for the heuristic and learned SVM gating networks.

does not rely on dynamic parameters, but it is completely data-driven and based on tactile sensors and force/torque sensors. As a result, our approach does not require a spatially calibrated model of the skin [18, 19]. This is a promising feature for robust control strategies that explicitly takes contacts into account.

Chapter 6

Learning Whole-Body Control using Tactile Sensing from Robot Skin (TUD)

Abstract

Whole-body control in presence of unknown obstacles is a challenging task. Unforeseen contacts with such obstacles can lead to poor tracking performance and potential physical damages. Hence, a whole-body control approach for future humanoid robots in unmodeled environments needs to take contact sensing into account. However, converting contact sensed with skin into physically well-understood quantities can be problematic as the exact position and strength of the contact would have to be converted into torque.

In this paper, we suggest an alternative approach that directly learns the mapping from both skin and joint state to the required torques needed for controlling the desired trajectory. We propose to learn such an inverse dynamics models with contacts using a "mixture of contacts" approach that exploits the linear superimposibility of contact forces. The learned model can accurately predict torques needed to compensate for the contact. As a result, trajectories with tactile contact can be executed more accurately even with low feedback gains and reduced risk of physical damage to both robot and environment.

We demonstrate on two different tasks on the humanoid robot *iCub* that this controller has a lower tracking error than classical alternatives.

6.1 Introduction

A fundamental problem for torque-controlled humanoid robots is to accurately model their dynamics in presence of contacts, e.g., during manipulation in clutter [33], whole-body movements [39] or ground contacts in locomotion [9]. Analytic models suffer from inaccurate dynamic parameters, unmodeled dynamics (e.g., friction, couplings, elasticities) and noisy sensor measurements. With contacts, the problem is even more challenging, because of discontinuities and additional non-linearities, which are difficult to model or estimate. Moreover, if contact locations are not fixed a priori or known with sufficient precision, small errors in the localization of the external force can substantially deteriorate the quality of the inverse dynamics [19].

Nevertheless, many modern control strategies like inverse dynamics control [22], computed torque control [74] or model predictive control approaches [56] rely on accurate dynamic models. With inaccurate dynamics models these control strategies can produce suboptimal policies, by not taking the external forces (caused by contacts) into account, and even damages to the hardware.

In contrast to classical techniques based on the identification of dynamics parameters [82, 59, 78], we propose a fully data-driven machine learning approach based on non-parametric models, where both the rigid body dynamics as well as the effect of external forces on the robot structure are learned directly from data collected on the real robot. The proposed model makes use of the raw sensory data and does not require a kinematic/dynamics calibration [82, 59, 78]: in particular, it does not need a spatially calibrated model of the skin [18]. As a non-linear model for the inverse dynamics we propose to learn a "mixtures of contacts" based on Gaussian Processes (GP).

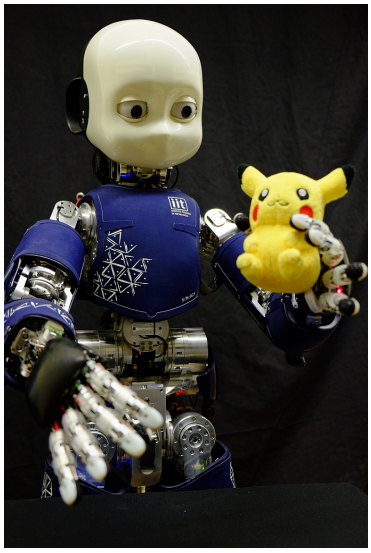


Figure 6.1: The humanoid robot *iCub* used in the experiments.

We evaluate our model learning approach on two different tasks using the arm of the *iCub* humanoid robot [55] (see Fig. 6.1) and compare against a state-of-the-art analytic modeling approach. In the first task, the learned inverse dynamics is used to compensate for an unexpected obstruction and minimize the tracking error. In the second task, we use the learned model on a controller designed to slide along an obstruction. The purpose of the sliding controller is to minimize the contact forces and therefore avoid to break the motors or the artificial tendons that actuate the joints in the case of unexpected contacts.

6.2 Inverse Dynamics

Without external contacts, the inverse dynamics of a robot with m degrees of freedom can be generally described by

$$\boldsymbol{\tau} = \underbrace{\boldsymbol{M}(\boldsymbol{q})\,\ddot{\boldsymbol{q}} + \boldsymbol{h}(\boldsymbol{q},\dot{\boldsymbol{q}})}_{\boldsymbol{\tau}_{\text{RBD}}} + \epsilon(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}})\,, \tag{6.1}$$

where $\boldsymbol{q} \in \mathbb{R}^m$, $\dot{\boldsymbol{q}} \in \mathbb{R}^m$ and $\ddot{\boldsymbol{q}} \in \mathbb{R}^m$ are the joint positions, velocities and accelerations, respectively, $\boldsymbol{M}(\boldsymbol{q}) \in \mathbb{R}^{m\times m}$ is the inertia matrix, and $\boldsymbol{h}(\boldsymbol{q},\dot{\boldsymbol{q}}) \in \mathbb{R}^{m\times m}$ is the matrix combining the contributions from Coriolis and centripetal, friction (viscous and static) and gravity forces:

$$\boldsymbol{h}(\boldsymbol{q},\dot{\boldsymbol{q}}) = C(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) + F_v\dot{\boldsymbol{q}} + F_s\,\mathrm{sgn}(\dot{\boldsymbol{q}})\,. \tag{6.2}$$

The term $\epsilon(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}})$ in (6.1) captures the errors of the model, such as unmodeled dynamics (e.g., elasticities and Stribeck friction), inaccuracies in the dynamic parameters (e.g., masses, inertia), vibrations, couplings, and sensor noise.

In presence of a set $\mathcal{C} = \{c_1 \dots c_n\}$ of contacts c_i between the robot and the environment, (6.1) becomes

$$\boldsymbol{\tau} = \underbrace{\boldsymbol{M}(\boldsymbol{q})\,\ddot{\boldsymbol{q}} + \boldsymbol{h}(\boldsymbol{q}, \dot{\boldsymbol{q}})}_{\boldsymbol{\tau}_{\text{RBD}}} + \epsilon(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) + \sum_{c_i \in \mathcal{C}} \boldsymbol{J}_{c_i}^{\top}(\boldsymbol{q})\,\boldsymbol{\gamma}_i\,, \tag{6.3}$$

where the last term accounts for the effect of the external wrenches (forces and moments) $\boldsymbol{\gamma}_i$ applied at the contact location c_i, and $\boldsymbol{J}_{c_i}(\boldsymbol{q})$ is the contact Jacobian[1].

6.2.1 Classical Model-based Approaches for Computing the Inverse Dynamics

Classical approaches for computing $\boldsymbol{\tau}$ or $\boldsymbol{\tau}_{\text{RBD}}$ rely on the dynamics model with known or identified kinematics and dynamics parameters [30]. The torques $\boldsymbol{\tau}_{\text{RBD}} = \boldsymbol{M}(\boldsymbol{q})\,\ddot{\boldsymbol{q}} + \boldsymbol{h}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ can be computed analytically through the rigid body dynamics model of the robot, a standard parametric description of the robot [24]. Conversely, the term $\epsilon(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$ is often neglected, or implicitly taken into account by considering a perturbation in the dynamics parameters of $\boldsymbol{\tau}_{\text{RBD}}$, which need to be identified accurately.

Although the parameter identification for industrial robots is relatively easy with exciting trajectories [64], the procedure for floating-base robots, such as humanoids, is not straightforward because of two main issues: The first issue is the generation of sufficiently large accelerations for the identification while maintaining the robot balance and the control of contacts. This issue was well explained by Yamane [83], who proposed a technique to identify the mass and the local COM of the links in a humanoid robot with fixed feet at the ground and slow joint trajectories. The second issue is the measurement of the external forces $\boldsymbol{\gamma}_i$ exerted on the robot. Note that it may not be straightforward to measure the external forces $\boldsymbol{\gamma}_i$, as it is not possible to cover the robot body with 6-axis force/torque sensors to measure the force exerted on every possible contact location c_i. Usually, such sensors are big, heavy and expensive, thus they are carefully placed where the external forces are critical for the main tasks, for example at the end-effectors for manipulation and at the feet for balancing. In such a case, it is possible to identify the dynamics parameters while balancing and walking without additional contacts [59]. When force/torque sensors are placed proximally, such as in the *iCub* arms [26], some of the dynamics parameters can be identified, but in absence of contacts [78].

When multiple contacts are exerted on the robot structure at locations other than the classical end-effectors, it is still possible to compute a precise inverse dynamics model, but this requires both pervasive joint torque sensing, such as in *Toro* [59], and additional force/torque and tactile sensing, such as in *iCub* [30]. Moreover, it requires the precise knowledge of the contact locations detected by the tactile sensors, which necessitates a spatial calibration of the skin [18]. This procedure is prone to errors, and it has been shown that small errors in the kinematics calibration of the taxels (i.e., the tactile units) can induce non-negligible errors in the estimation of the contact forces [19].

[1] The contact location c_i is not necessarily fixed, as the contacts may occur on the whole robotic structure and not exclusively at the end-effectors. In such a case, the contact location, if not known a priori, must be estimated, typically through distributed tactile sensors. To compute the contact Jacobian, we need the position of the contact point with respect to the reference frame of the link [26]. Such a knowledge requires a kinematic calibration of the skin, as explained in [18].

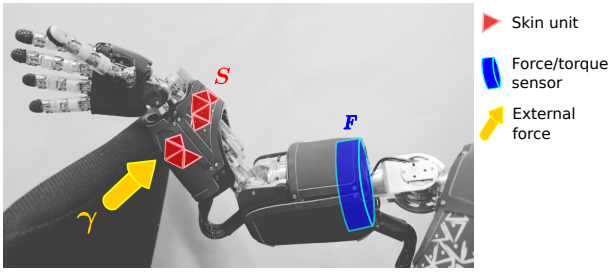


Figure 6.2: Illustration of the force/torque and tactile sensors involved during a contact of the robot arm with the environment.

Overall, these approaches have three main limitations: First, since they are model-based, it is difficult to add details about couplings, elasticity, friction and other nonlinear dynamics, which would be required for high accuracy; Second, the performance of the data-driven identification strongly depends on the experimental setting (with/without contacts) and the exciting trajectories [64]; Third, they make strong assumptions in order to handle contacts.

6.2.2 Learning the Inverse Dynamics

An alternative and appealing approach to model-based dynamics computation is to use machine learning methods to learn the dynamics model of a robot [58, 80, 16]. Without the need to compensate for inaccurate dynamics parameters and accumulated errors, a learned dynamics model can improve the tracking and control performance of a robot, as shown in [57] for an industrial manipulator. The clear advantage of learning the inverse dynamics is that we can overcome the limitations of the aforementioned approaches: difficulty in modeling complex nonlinear dynamics, impossibility to generate suitable exciting trajectories, restrictive assumptions regarding contacts and sensors, prior accurate kinematics calibration of the tactile sensors.

The inclusion of multiple contact models in the dynamics highlights two main problems: First, switching from a no-contact model to a contact-model requires to observe the system state and to model a discontinuous function [76]. Second, switching between different contacts $c_i \in \mathcal{C}$ must be properly handled.

Here, we provide a formulation to this problem, and we show that it is possible to learn the inverse dynamics model of the robot by means of proximal force/torque measurements $\boldsymbol{F}$ and distributed tactile sensors $\boldsymbol{S}$ such that:

$$\boldsymbol{\tau} = \boldsymbol{\tau}_{\text{IDM}}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}, \boldsymbol{S}, \boldsymbol{F}) \,. \tag{6.4}$$

This solution enables a fast and accurate prediction of joint torques in situations when the robot is in contact with no, one or even multiple simultaneous contacts, detected by a tactile skin. The estimation does not rely on dynamic parameters or parametric models, but it is completely data driven: Tactile sensors provide information about the contact locations (without requiring a spatially calibrated model of the skin [18]), while force/torque sensors provide information about the wrenches perceived by the robotic structure.

We detail our proposed model and its learning procedure in the following section.

6.3 Control with Tactile Sensing

In this section, we present our proposed approach to learning inverse dynamics with contacts. We first formalize the problem as learning a mixture-of-experts model. Then we detail how to implement Gaussian processes as the corresponding experts.

6.3.1 Learning a Mixture-of-Contacts

When learning the inverse dynamics with contacts ((6.3)), we assume that the (contact-free) inverse dynamics from (6.1) can be computed precisely, either from an analytical model or from a learned model [57]. In our experiments, we employ a learned GP model for this purpose. The reason for this choice are the unmodeled dynamics $\epsilon\left(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}\right)$, which introduce substantial errors even in absence of contacts. As a result of this contact-free inverse dynamics, only the model of the additional term of the external forces $\sum_{i \in \mathcal{C}} \boldsymbol{J}_i(\boldsymbol{q})^\top \boldsymbol{\gamma}_i$ has to be learned. In this paper, we consider a robot that is provided with skin measurements $\boldsymbol{S}$ from the tactile sensors, force measurements $\boldsymbol{F}$ from the force torque sensors (FTS) and the applied torques $\boldsymbol{\tau}$. A visual representation of these relevant components is shown in Fig. 6.2. Predicting the external forces $\sum_{i \in \mathcal{C}} \boldsymbol{J}_i(\boldsymbol{q})^\top \boldsymbol{\gamma}_i$ can be formalized as the regression task

$$\boldsymbol{y} = f(\boldsymbol{x}) + \epsilon\,, \tag{6.5}$$

where $\boldsymbol{y} = \sum_{i \in \mathcal{C}} \boldsymbol{J}_i(\boldsymbol{q})^\top \boldsymbol{\gamma}_i$ and $\boldsymbol{x} = [\boldsymbol{q}, \boldsymbol{S}, \boldsymbol{F}]$ are the inputs. Additionally, ϵ is an i.i.d. Gaussian measurement noise with mean 0 and variance σ_n. Therefore, our regression problem is phrased as

$$\boldsymbol{y} = \sum_{i \in \mathcal{C}} \boldsymbol{J}_i(\boldsymbol{q})^\top \boldsymbol{\gamma}_i = f([\boldsymbol{q}, \boldsymbol{S}, \boldsymbol{F}]) + \epsilon\,. \tag{6.6}$$

It is necessary to consider the skin as an input $\boldsymbol{S}$ since contacts with different parts of the body lead to different effects in the dynamics. Intuitively, $\boldsymbol{S}$ is required to identify the position of the contact. The force/torque measurements $\boldsymbol{F}$ could be avoided if we were interested in learning contacts that do not change between training and test time, which would restrict us to dealing with static objects, such as a rigid floor, walls or stationary obstacles. However, as this assumption is limiting, we include the force/torque measurements $\boldsymbol{F}$ in our model.

The resulting regression of (6.6) is a highly complex task, due to the extremely high-dimensional space of the input $\boldsymbol{x} \in \mathcal{X}$ (the skin measurements $\boldsymbol{S}$ alone account for hundreds of dimensions) and nonlinearity. We tackle this problem by rephrasing it as a problem of learning a mixture-of-experts model ("mixture of contacts" in our case). With this model, we decompose (6.6) as

$$\sum_{i \in \mathcal{C}} \boldsymbol{J}_i(\boldsymbol{q})^\top \boldsymbol{\gamma}_i = \sum_{j \in \mathcal{J}} f_j([\boldsymbol{q}, \boldsymbol{F}]) + \epsilon\,, \tag{6.7}$$

where $\mathcal{J}$ is the set of active experts f_j. Note that the skin input $\boldsymbol{S}$ is no longer explicitly part of the inputs of the experts. Hence, each single expert f_j is now sufficiently low-dimensional to be modeled independently, but at the same time the possibility of summing the contribution of each contact allows to account for complex behaviors. As single expert f_j we propose to

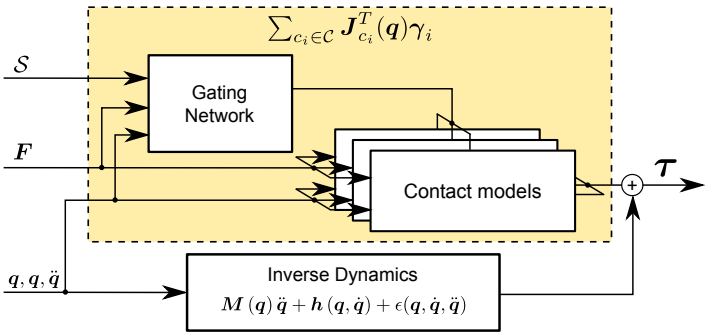


Figure 6.3: Our approach extends existing inverse dynamics without contacts by learning many contact models which serve as correction terms under different contacts type. The decision of which contact model to activate is taken by a gating network based on the skin measurements $\boldsymbol{S}$, the force torque sensors $\boldsymbol{F}$ and the current state $\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}$.

use Gaussian processes mapping $[\boldsymbol{q}, \boldsymbol{F}] \mapsto \boldsymbol{J}_j(\boldsymbol{q})^\top \boldsymbol{\gamma}_j$. Detailed information regarding the GP models and their training are given in the next subsection. The purpose of the gating network is then to select the experts that are currently active and to add their contributions. An illustration of our approach is shown in Fig. 6.3. For mixture-of-experts models it is required to design a suitable gating network that activates the relevant experts. In our case, this gating network can be considered a classifier $\mathcal{J} = g(\boldsymbol{q}, \boldsymbol{S}, \boldsymbol{F})$ that selects which contact is currently ongoing. For simple tasks, this gating network can be designed using heuristics (e.g., using thresholds on the activation of the tactile sensors). Alternatively, for more complex systems an approach based on machine learning is more suitable.

6.3.2 Gaussian Processes as Expert Models

Gaussian Processes [66] are a state-of-the-art regression method. They have been used in robotics to learn dynamics models [16] and for control [17]. In the context of this paper, a GP is a distribution over inverse dynamics models

$$f \sim \mathcal{GP}\left(m_f, k_f\right) , \tag{6.8}$$

fully defined by a prior mean m_f and a covariance function k_f. In our experiments, we choose as prior mean $m_f \equiv \boldsymbol{\tau}_{\text{RBD}}$ and as covariance function k_f the squared exponential with automatic relevance determination and Gaussian noise:

$$k(\boldsymbol{x}_p, \boldsymbol{x}_q) = \sigma_f^2 \exp\left(-\tfrac{1}{2}(\boldsymbol{x}_p - \boldsymbol{x}_q)^T \boldsymbol{\Lambda}^{-1}(\boldsymbol{x}_p - \boldsymbol{x}_q)\right) + \sigma_w^2 \delta_{pq} \tag{6.9}$$

where $\boldsymbol{\Lambda} = \text{diag}([l_1^2, ..., l_D^2])$ and δ_{pq} is the Kronecker delta (which is one if $p = q$ and zero otherwise). Here, l_i are the characteristic length-scales, σ_f^2 is the variance of the latent function $f(\cdot)$ and σ_w^2 the noise variance. In our experiments, when learning contact models, the input is defined as $\boldsymbol{x} = [\boldsymbol{q}, \boldsymbol{F}]$ and the output (observations) is $\boldsymbol{y} = \boldsymbol{\tau}$ are the torques. Hence, given n training inputs $\boldsymbol{X} = [\boldsymbol{x}_1, ..., \boldsymbol{x}_n]$ and corresponding training targets $\boldsymbol{y} = [y_1, ..., y_n]$, we define the training data set $\mathbb{D} = \{\boldsymbol{X}, \boldsymbol{y}\}$. Training the GP corresponds to finding good hyperparameters $\boldsymbol{x} = [l_i, \sigma_f, \sigma_w]$, which is done by the standard procedure of maximizing the marginal likelihood [66].

The GP yields the predictive distribution over torques for a new input $\boldsymbol{x}_* = [\boldsymbol{q}_*, \boldsymbol{F}_*]$

$$p(\boldsymbol{y}|\mathbb{D}, \boldsymbol{x}_*) = \mathcal{N}\left(\mu(\boldsymbol{x}_*), \sigma^2(\boldsymbol{x}_*)\right) ,$$

where the mean $\mu(\boldsymbol{x}_*)$ and the variance $\sigma^2(\boldsymbol{x}_*)$ are

$$\mu(\boldsymbol{x}_*) = \boldsymbol{k}_*^T \boldsymbol{K}^{-1} \boldsymbol{y} , \quad \sigma^2(\boldsymbol{x}_*) = k_{**} - \boldsymbol{k}_*^T \boldsymbol{K}^{-1} \boldsymbol{k}_* .$$

The entries of the matrix $\boldsymbol{K}$ are $K_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, and we define $k_{**} = k(\boldsymbol{x}, \boldsymbol{x})$ and $\boldsymbol{k}_* = k(\boldsymbol{X}, \boldsymbol{x})$.

6.3.3 Controlling the Contacts

In the case of no contacts $\mathcal{C} = \{0\}$ we can define the Task-space Nonlinear Feedforward Control:

$$\boldsymbol{u} = \boldsymbol{\tau}_{\text{RBD}} , \tag{6.10}$$

where the $\boldsymbol{\tau}_{\text{RBD}}$ is computed from the rigid body inverse dynamics (or a learned model of it). Often an additional PD feedback controller is added to compensate for noise and inaccuracies in the dynamics, such that

$$\boldsymbol{u} = \boldsymbol{\tau}_{\text{RBD}} + \underbrace{K_P \left(\boldsymbol{q}^{\text{des}} - \boldsymbol{q}\right) + K_D \left(\dot{\boldsymbol{q}}^{\text{des}} - \dot{\boldsymbol{q}}\right)}_{\boldsymbol{\tau}_{PD}} . \tag{6.11}$$

Intuitively, the magnitude of the torques contribution from the PD controller $\boldsymbol{\tau}_{PD}$ can be used to measure the goodness of our inverse dynamics model. Accurate inverse dynamics model will only need small corrections by the feedback controller, while inaccurate models will rely more heavily on it. In case of inaccurate models increasing the PD gains can still lead to acceptable tracking performance at the expense of safety. However, with unforeseen obstacles, high gains can lead to both damages to the robot's hardware and the obstacle itself.

6.4 Experimental Results

In this section we present the experiments we will conduct. Two different tracking tasks in presence of external contacts will be investigated. First, we will demonstrate that a controller using a learned inverse dynamics model can be used to compensate for contact forces and that the tracking performance of a tracking controller will be improved. In a second experiment, we will demonstrate that the same inverse dynamics model can also be used to avoid an obstacle and gently slide along it.

6.4.1 Experimental Setting

Both experimental evaluations are performed on a real *iCub* humanoid robot [55]. The *iCub* possess 53 degrees of freedom and is 104 cm tall for 24 kg of weight. Four 6-axis force/torque sensors placed are proximally in the middle of legs and arms. Additionally, artificial skin

consisting of more than 2000 tactile sensors are mounted on the robot covers [10]. In our experiments, we control 5 DoF of the *iCub* arm: shoulder pitch, roll and jaw, elbow and wrist pronosupination. Therefore $\boldsymbol{q} \in \mathbb{R}^5$, $\dot{\boldsymbol{q}} \in \mathbb{R}^5$, $\ddot{\boldsymbol{q}} \in \mathbb{R}^5$, $\boldsymbol{\tau} \in \mathbb{R}^5$ and $\boldsymbol{F} \in \mathbb{R}^3$ resulting in learning the mapping $\boldsymbol{x} \in \mathbb{R}^{18} \mapsto \boldsymbol{y} \in \mathbb{R}^5$. The skin input $\boldsymbol{S}$ from the forearm consists of 270 sensors.

6.4.2 Pushing Obstacles

For classical controllers, when an obstruction occur, the rigid body inverse dynamics does not account for this variation. As a result, the tracking error increase and the contribution of the PD feedback controller increases to compensate for this tracking error. In this scenario we demonstrate that it is possible to use a learned model to improve the tracking accuracy when unforeseen and unknown obstruction are encountered along the path. Additionally, we show that such learned dynamics allows to reduce the PD gains to achieve an increased safety, without loss in tracking accuracy.

We first consider the *iCub* following a pre-defined trajectory with the left arm. Following, we repeat the same trajectory, but this time with an unforeseen obstruction.

To compare the performance of our learned inverse dynamics we first analyze the tracking error introduced by the obstruction.

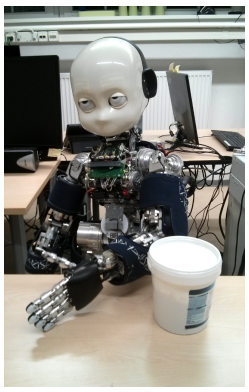


Figure 6.4: **Pushing Obstacles.**

Bibliography

[1] Jonathan Baxter. A model of inductive bias learning. *Journal of Artificial Intelligence Research*, 12:149–198, 2000.

[2] Yasemin Bekiroglu, Renaud Detry, and Danica Kragic. Learning tactile characterizations of object- and pose-specific grasps. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011.

[3] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, USA, 2006.

[4] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[5] J. Bohg, M. Johnson-Roberson, B. León, J. Felip, X. Gratal, N. Bergström, D. Kragic, and A. Morales. Mind the gap - robotic grasping under incomplete observation. In *proceedings of International Conference on Robotics and Automation*, pages 686–693, 2011.

[6] J. Bohg, A. Morales, T. Asfour, and D. Kragic. Data-driven grasp synthesis - a survey. *IEEE Transactions on Robotics*, accepted.

[7] A. Boularias, O. Kroemer, and J. Peters. Learning robot grasping from 3d images with markov random fields. In *IEEE/RSJ International Conference on Intelligent Robot Systems (IROS)*, 2011.

[8] R. Calandra, S. Ivaldi, M. Deisenroth, E. Rueckert, and J. Peters. Learning inverse dynamics models with contacts. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, Seattle, USA, 2015.

[9] Roberto Calandra, Nakul Gopalan, André Seyfarth, Jan Peters, and Marc Peter Deisenroth. Bayesian gait optimization for bipedal locomotion. In *LION8*, 2014.

[10] Giorgio Cannata, M. Maggiali, G. Metta, and G. Sandini. An embedded artificial skin for humanoid robots. In *MFI*, 2008.

[11] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A Library for Support Vector Machines. *ACM Trans. Intell. Syst. Technol.*, 2(3):27:1–27, 2011.

[12] Zhaopeng Chen, Neal Y. Lii, Thomas Wimboeck, Shaowei Fan, Minghe Jin, Christoph Borst, and Hong Liu. Experimental study on impedance control for the five-finger dexterous robot hand dlr-hit ii. In *IROS*, pages 5867–5874. IEEE, 2010.

[13] Hao Dang and Peter K. Allen. Learning grasp stability. In *ICRA*, pages 2392–2397. IEEE, 2012.

[14] Hal Daume. Bayesian multitask learning with latent hierarchies. In *Proc. Conf. on Uncertainty in Artificial Intelligence*, 2009.

[15] Andrea d'Avella, Philippe Saltiel, and Emilio Bizzi. Combinations of Muscle Synergies in the Construction of a Natural Motor Behavior. *Nature*, 6(3):300–308, March 2003.

[16] Marc Peter Deisenroth, Roberto Calandra, André Seyfarth, and Jan Peters. Toward fast policy search for learning legged locomotion. In *IROS*, 2012.

[17] M.P. Deisenroth, D. Fox, and C. Rasmussen. Gaussian Processes for Data-Efficient Learning in Robotics and Control. *IEEE TPAMI*, 37(2):408–423, 2015.

[18] Andrea Del Prete, Simone Denei, Lorenzo Natale, Fulvio Mastrogiovanni, Francesco Nori, Giorgio Cannata, and Giorgio Metta. Skin spatial calibration using force/torque measurements. In *IROS*, 2011.

[19] Andrea Del Prete, Francesco Nori, Giorgio Metta, and Lorenzo Natale. Control of contact forces: The role of tactile feedback for contact localization. In *IROS*, 2012.

[20] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

[21] Renaud Detry, Carl Henrik Ek, Marianna Madry, Justus Piater, and Danica Kragic. Generalizing grasps across partly similar objects. In *IEEE International Conference on Robotics and Automation*, 2012.

[22] Tom Erez and Emanuel Todorov. Trajectory optimization for domains with contacts using inverse dynamics. In *IROS*, 2012.

[23] Theodoros Evgeniou and Massimiliano Pontil. Regularized multi-task learning. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2004.

[24] Roy Featherstone and D.E. Orin. Dynamics. In *Springer Handbook of Robotics*, pages 35–65. Springer Berlin Heidelberg, 2008.

[25] Denis Forte, Andrej Gams, Jun Morimoto, and Aleš Ude. On-line motion synthesis and adaptation using a trajectory database. *Robotics and Autonomous Systems*, 60(10):1327–1339, 2012.

[26] M. Fumagalli, S. Ivaldi, M. Randazzo, L. Natale, G. Metta, G. Sandini, and F. Nori. Force feedback exploiting tactile and proximal force/torque sensing - theory and implementation on the humanoid robot icub. *Autonomous Robots*, 33(4):381–398, 2012.

[27] James J. Gibson. *The Ecological Approach To Visual Perception*. Lawrence Erlbaum Associates, new edition edition, September 1986.

[28] Alexander Herzog, Peter Pastor, Mrinal Kalakrishnan, Ludovic Righetti, Jeannette Bohg, Tamim Asfour, and Stefan Schaal. Learning of grasp selection based on shape-templates. *Autonomous Robots*, 36(1-2):51–65, 2014.

[29] A. Ijspeert and S. Schaal. Learning Attractor Landscapes for Learning Motor Primitives. In *Advances in Neural Information Processing Systems 15*, (NIPS). MIT Press, Cambridge, MA, 2003.

[30] S. Ivaldi, M. Fumagalli, M. Randazzo, F. Nori, G. Metta, and G. Sandini. Computing robot internal/external wrenches by means of inertial, tactile and F/T sensors: theory and implementation on the iCub. In *HUMANOIDS*, 2011.

[31] S. Ivaldi, J. Peters, V. Padois, and F. Nori. Tools for simulating humanoid robot dynamics: a survey based on user feedback. In *HUMANOIDS*, 2014.

[32] A. Jain, M.D. Killpack, A. Edsinger, and C.C. Kemp. Reaching in clutter with whole-arm tactile sensing. *IJRR*, 32(4):458–482, 2013.

[33] A. Jain, M.D. Killpack, A. Edsinger, and C.C. Kemp. Reaching in clutter with whole-arm tactile sensing. *The Int. Journ. of Robotics Research*, 32(4):458–482, 2013.

[34] Tony Jebara and Risi Kondor. Bhattacharyya expected likelihood kernels. In *COLT*, volume 2777 of *Lecture Notes in Computer Science*, pages 57–71. Springer, 2003.

[35] Tony Jebara, Risi Kondor, and Andrew Howard. Probability product kernels. *J. Mach. Learn. Res.*, 5:819–844, December 2004.

[36] Robert Jenssen, Jose C. Principe, Deniz Erdogmus, and Torbjorn Eltoft. The cauchy-schwarz divergence and parzen windowing: Connections to graph theory and mercer kernels. *Journal of the Franklin Institute*, 343(6):614–629, 2006.

[37] Yun Jiang, Marcus Lim, Changxi Zheng, and Ashutosh Saxena. Learning to place new objects in a scene. *I. J. Robotic Res.*, 31(9):1021–1043, 2012.

[38] Shuuji Kajita and Bernard Espiau. Legged robots. In *Handbook of Robotics*, pages 361–389. Springer, 2008.

[39] Shuuji Kajita and Bernard Espiau. Legged robots. In Bruno Siciliano and Oussama Khatib, editors, *Springer Handbook of Robotics*, pages 361–389. Springer Berlin Heidelberg, 2008.

[40] M. Khansari-Zadeh and A. Billard. Learning Stable Non-Linear Dynamical Systems with Gaussian Mixture Models. *IEEE Transaction on Robotics*, 2011.

[41] J. Kober, E. Oztop, and J. Peters. Reinforcement Learning to adjust Robot Movements to New Situations. In *Proceedings of the Robotics: Science and Systems Conference (RSS)*, 2010.

[42] J. Kober and J. Peters. Policy Search for Motor Primitives in Robotics. *Machine Learning*, pages 1–33, 2010.

[43] Marek Sewer Kopicki, Sebastian Zurek, Rustam Stolkin, Thomas Morwald, and Jeremy L. Wyatt. Learning to predict how rigid objects behave under simple manipulation. In *ICRA*, pages 5722–5729. IEEE, 2011.

[44] Hema Koppula and Ashutosh Saxena. Anticipating human activities using object affordances for reactive robotic response. In *RSS*, 2013.

[45] O. Kroemer and J. Peters. Predicting object interactions from contact distributions. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2014.

[46] O. Kroemer, E. Ugur, E. Oztop, and J. Peters. A kernel-based approach to direct action perception. In *International Conference on Robotics and Automation (ICRA)*, 2012.

[47] Oliver Kroemer, H Ben Amor, Marco Ewerton, and Jan Peters. Point cloud completion using extrusions. In *Humanoid Robots (Humanoids), 2012 12th IEEE-RAS International Conference on*, pages 680–685. IEEE, 2012.

[48] Johannes Kulick, Tobias Lang, Marc Toussaint, and Manuel Lopes. Active Learning for Teaching a Robot Grounded Relational Symbols. In *International Joint Conference on Artificial Intelligence*, Beijing, China, 2013.

[49] Abhishek Kumar and Hal Daume. Learning task grouping and overlap in multi-task learning. In *Proceedings of the 29th international conference on Machine Learning*, 2012.

[50] A. Kupcsik, M. P. Deisenroth, J. Peters, and G. Neumann. Data-Efficient Contextual Policy Search for Robot Movement Skills. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2013.

[51] C.H. Lampert and J. Peters. Real-time detection of colored objects in multiple camera streams with off-the-shelf hardware components. *Journal of Real-Time Image Processing*, 2012.

[52] Alessandro Lazaric and Mohammad Ghavamzadeh. Bayesian multi-task reinforcement learning. In *ICML '10 Proceedings of the 27th international conference on Machine Learning*, 2010.

[53] L. Montesano, M. Lopes, A. Bernardino, and J. Santos-Victor. Modeling affordances using bayesian networks. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 4102–4107, Oct 2007.

[54] K. Mülling, J. Kober, and J. Peters. A Biomimetic Approach to Robot Table Tennis. *Adaptive Behavior Journal*, (5), 2011.

[55] L. Natale, F. Nori, G. Metta, M. Fumagalli, S. Ivaldi, U. Pattacini, M. Randazzo, A. Schmitz, and G. G. Sandini. The iCub platform: a tool for studying intrinsically motivated learning. In *Intrinsically motivated learning in natural and artificial systems*. Springer, 2013.

[56] M. Naveau, J. Carpentier, S. Barthelemy, O. Stasse, and P. Soueres. METAPOD - Template META-PrOgramming applied to Dynamics: CoP-CoM trajectories filtering. In *HUMANOIDS*, 2014.

[57] Duy Nguyen-Tuong and Jan Peters. Model learning for robot control: a survey. *Cognitive processing*, 12(4):319–340, 2011.

[58] Duy Nguyen-Tuong, Jan Peters, and Matthias Seeger. Local Gaussian process regression for real time online model learning. In *NIPS*, 2008.

[59] Y. Ogawa, G. Venture, and C. Ott. Dynamic parameters identification of a humanoid robot using joint torque sensors and/or contact forces. In *HUMANOIDS*, 2014.

[60] A. Paraschos, C. Daniel, J. Peters, and G Neumann. Probabilistic movement primitives. In *Advances in Neural Information Processing Systems (NIPS), Cambridge, MA: MIT Press.*, 2013.

[61] A. Paraschos, G Neumann, and J. Peters. A probabilistic approach to robot trajectory generation. In *Proceedings of the International Conference on Humanoid Robots (HUMANOIDS)*, 2013.

[62] Alexandre Passos, Piyush Rai, Jacques Wainer, and Hal Daume. Flexible modeling of latent task structures in multitask learning. In *In Proceedings of International Conference on Machine Learning*, 2012.

[63] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal. Learning and Generalization of Motor Skills by Learning from Demonstration. In *International Conference on Robotics and Automation (ICRA)*, 2009.

[64] N. Pedrocchi, E. Villagrossi, F. Vicentini, and L. Tosatti. Robot-dyanmic calibration improvement by local identification. In *ICRA*, 2014.

[65] P. Rai and H. Daume. Infinite predictor subspace models for multitask learning. In *Int. Conf. on Artificial Intelligence and Statistics*, 2010.

[66] Carl E. Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[67] Benjamin Rosman and Subramanian Ramamoorthy. Learning spatial relationships between objects. *I. J. Robotic Res.*, 30(11):1328–1342, 2011.

[68] L. Rozo, S. Calinon, D. G. Caldwell, P. Jimenez, and C. Torras. Learning collaborative impedance-based robot behaviors. In *AAAI Conference on Artificial Intelligence*, 2013.

[69] E. Rueckert, J. Mundo, A. Paraschos, J. Peters, and G. Neumann. Extracting low-dimensional control variables for movement primitives. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pages 1–9, Seattle, USA, May 26-30 2015.

[70] Radu Bogdan Rusu and Steve Cousins. 3d is here: Point cloud library (pcl). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[71] Paul Ruvolo and Eric Eaton. Online multi-task learning via sparse dictionary optimization. In *Twenty-Eighth AAAI Conference on Artificial Intelligence (AAAI-14)*, July 2014.

[72] E. Sahin, M. Cakmak, M. R. Dogar, E. Ugur, and G. Ucoluk. To afford or not to afford: A new formalization of affordances toward affordance-based robot control. *Adaptive Behavior*, 15(4):447–472, December 2007.

[73] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. The MIT Press, 1st edition, 2001.

[74] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo. *Robotics: Modelling, Planning and Control*. Springer, 2009.

[75] K. Sjoo and P. Jensfelt. Learning spatial relations from functional simulation. In *Intelligent Robots and Systems (IROS)*, pages 1513–1519, Sept 2011.

[76] Marc Toussaint and Sethu Vijayakumar. Learning discontinuities with products-of-sigmoids for switching between local models. In *ICML*, 2005.

[77] Silvio Traversaro, Andrea Del Prete, Riccardo Muradore, Lorenzo Natale, and Francesco Nori. Inertial parameter identification including friction and motor dynamics. *HUMANOIDS*, 2013.

[78] Silvio Traversaro, Andrea Del Prete, Riccardo Muradore, Lorenzo Natale, and Francesco Nori. Inertial parameter identification including friction and motor dynamics. *IEEE-RAS International Conference on Humanoid Robots (Humanoid13), Atlanta, USA*, 2013.

[79] A. Ude, A. Gams, T. Asfour, and J. Morimoto. Task-specific generalization of discrete and periodic dynamic movement primitives. *Trans. Rob.*, (5), 2010.

[80] Sethu Vijayakumar and Stefan Schaal. Locally weighted projection regression: Incremental real time learning in high dimensional space. In *ICML*, 2000.

[81] Ya Xue, Xuejun Liao, Lawrence Carin, and Balaji Krishnapuram. Multi-task learning for classification with dirichlet process priors. *Journal of Machine Learning Research*, 8:2007, 2007.

[82] K. Yamane. Practical kinematic and dynamic calibration methods for force-controlled humanoid robots. In *HUMANOIDS*, 2011.

[83] K. Yamane. Practical kinematic and dynamic calibration methods for force-controlled humanoid robots. In *HUMANOIDS*, 2011.

[84] Kai Yu, Voker Tresp, and Anton Schwaighofer. Learning gaussian processes from multiple tasks. In *ICML 2005 Proceedings of the 22nd international conference on Machine learning*, pages 1012 – 1019, 2005.