FP7-600716

Whole-Body Compliant Dynamical Contacts in Cognitive Humanoids

D4.1
Learning approaches for operational space control with contacts.

Editor(s)	Jan Peters[1,2] and Elmar Rueckert[1]
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. 29, 2016
EC Distribution:	Consortium
Project Number:	600716
Project Title:	Whole-Body Compliant Dynamical Contacts in Cognitive Humanoids

Title of Deliverable:	Learning approaches for operational space control with contacts.
Date of delivery to the EC:	29/2/2016

Workpackage responsible for the Deliverable	WP4
Editor(s):	Jan Peters and Elmar Rueckert
Contributor(s):	Alex Paraschos, Daniel Tanneberg, Jan Peters and Elmar Rueckert (TUD)
Reviewer(s):	
Approved by:	All Partners

Abstract	The scope of the current deliverable is to present the results on solving operational space control problems in humanoid robots.
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. 29	Final	Elmar Rueckert

Table of Contents

Chapter 1

Introduction

This deliverable presents results of task T4.1 and of task T4.2 at the end of the thrid year. The achieved results are briefly discussed with respect to the task description from the Technical Annex.

T4.1 Generalizing and Improving Elementary Tasks with Contacts. Analytical models that include physical robot-environment contact are frequently not sufficiently accurate for gentle and dextrous interaction with the environment. Hence, there is a strong need for improved models that enable more versatile predictions (e.g., of the next state or a force required to accomplish an acceleration) for whole-body contact. Core problems in these contexts are under-modelled nonlinearities (e.g., hydraulic tubes, cable drives, etc.) as well as that the various contact situations have substantially different interactive behaviour. As there may be unforeseeably many possible scenarios, real-time regression techniques may be one of the most reasonable approaches for such kind of problems. However, we avoid the classical problem of learning from scratch but rather learn the part of the dynamics that has not been captured by the preceding classical system identification techniques. The resulting model is obviously more accurate than the original and its training more efficient. As the type of contact may decide upon the type of interaction, it is an essential feature. However, it is not directly accessible but a latent cause for a change in behaviour and needs to be inferred online in real-time. Hence, new real-time regression techniques need to be developed particularly for a multi-contact situation where multiple model classes exist and generalization between different latent contacts. Here is a list of achievements for this task:

- TUD learned predictive models of joint angles and task-space forces in a probabilistic control framework. Early results were presented at the end of year two. In particular, TUD developed a model-free control method that can be trained from demonstrations and generates time-varying feedback control gains that reproduces the demonstrations. In this approach a joint distribution over states, sensory feedback (e.g., measured joint torques or contact forces) and controls is learned. In conditioning on the current state the next-state control-law can be computed in closed form approximating the true forward dynamics through local linearizations given the demonstrations. During year three, TUD refined this approach and evaluated the model-free ProMP method on the humanoid robot iCub in lifting objects. The model could generalize to different object weights and grasp locations. A paper was published and presented at the IEEE/RSJ Conference on

Intelligent Robots and Systems (IROS) in Hamburg, Germany 2015 [1]. This work is discussed in Chapter 3.

T4.2 Inferring the Operational Space and Appropriate Controls with Multiple Contacts. Operational space control (OSC) is among the most general frameworks for phrasing control tasks. It allows formulating the task in its own inherent space instead of using a more action-oriented space such as the joint-space. Current approaches use human insight for finding the appropriate task space and analytical physical models in order to compute the appropriate controls that realize the task (Nakanishi et al., 2008). Clearly model errors can have devastating impacts on the performance of operational space control laws and, hence, learning approaches have been introduced (Peters & Schaal, 2008) (Salan et al., 2010). As these learning approaches do not yet include contact, the extension to a multi-contact scenario will be essential. However, this step is neither straightforward nor incremental. Only by relying on the latent contact inference engine from T4.1, the new Learning OSC module will be able to deal with learning multiple different OSCs for different scenarios.

An additional trouble arises from finding the appropriate space for a task. This step is known to be very difficult, as here the latent manifold of the task will need to be considered. We will present a machine learning approach that will recover the latent task space from the data and models provided by WP2. Here is a list of achievements for this task:

- For controlling high-dimensional robots, most stochastic optimal control algorithms use approximations of the system dynamics and of the cost function (e.g., using linearizations and Taylor expansions). These approximations are typically only locally correct, which might cause instabilities in the greedy policy updates, lead to oscillations or the algorithms diverge. To overcome these drawbacks, TUD added a regularization term to the cost function that punishes large policy update steps in the trajectory optimization procedure. TUD applied this concept to the Approximate Inference Control method (AICO), where the resulting algorithm guarantees convergence for uninformative initial solutions without complex hand-tuning of learning rates. The new algorithm was evaluated on two simulated robotic platforms. A robot arm with five joints was used for reaching multiple targets while keeping the roll angle constant. On the humanoid robot Nao, we show how complex skills like reaching and balancing can be inferred from desired center of gravity or end effector coordinates. This work was presented at the International Conference on Humanoid Robots (HUMANOIDS) in Madrid, Spain in 2014 [3]. The stochastic optimal control algorithm is discussed in Chapter 4.

- In an ongoing student project, TUD currently evaluates this approach on a real system (the Nao robot). A journal paper is in progress of writing.

- Another source of inspiration for novel operational space control algorithms is drawn from observations of how animals solve planning tasks. In particular, TUD investigated how spiking neural networks can be used to learn a mapping between joint and task space and utilized this mapping in a temporal model for movement planning. A paper demonstrating for the first time how a spiking neural network architecture can implement probabilistic planning through local reward modulated synaptic plasticity rules was published the journal of Scientific Reports [2]. The main contribution is a theoretical model

with implications for neuroscience studies on path planning in the hippocampus and robot controller implemented on neuromorphic hardware. Optimal learning rules were derived based on probabilistic inference and links to the widely used machine learning techniques expectation maximization and policy search were established.

As computational model for hippocampal sweeps during maze navigation tasks, the model suggests that contextual information about rewarding goal locations or obstacles modulate not only the end point of a path (as done in state-of-the-art attractor models) but the complete movement trajectory.

For robotics, the resulting neural network offers the foundation for future neuromorphic hardware implementations that can be used in dynamic human robot co-worker environments. In this challenging domain large sensory streams from cameras and tactile skins have to processed and stochastic decisions have to be computed based on noisy or partial observations taking future expectations about human motion into account. In first experiments, it is shown that the neural model can be trained to represent and generate multimodal movement plans that avoid obstacles in a real robot arm reaching task in real-time. This work is discussed in Chapter 5.

- In a student's Master thesis, TUD refined the above spiking neural network model to a deep neural network architecture. In using factorized population codes, the model could generalize to high-dimensional robot systems (i.e., a robot with more than six joints). TUD demonstrated that both, the state transition model and the inverse kinematic model can be learned from human demonstrations using kinesthetic teaching. For that purpose TUD derived a spike dependent version of contrastive divergence. The models are learned and evaluated on a KUKA lightweight arm in simulation and on the real robot by solving target reaching and obstacle avoidance tasks. A paper on these results is currently in progress of writing.

Chapter 2

Executive Summary

In this deliverable, three studies relevant for improving operational space control with contacts are presented. The first work demonstrates how model-free probabilistic movement primitives can be learned from human demonstrations and generalize to different task-space forces. In a second study, TUD demonstrated how a simple regularization term can greatly improve the convergence properties of incremental stochastic optimal control methods. This work was evaluated in simulations and is currently tested on real humanoid robots. In the last chapter of this deliverable we present an interesting control approach using spiking neural networks. The approach does not aim at solving the challenging control problems within CoDyCo but rather provides an alternative with great potentials for future research. As such TUD is investigating how deep neural network variants of this model can be used for controlling real robots in high-dimensional joint and task spaces.

Chapter 3

Model-Free Probabilistic Movement Primitives for Physical Interaction (TUD)

Model-Free Probabilistic Movement Primitives for Physical Interaction

Alexandros Paraschos[1], Elmar Rueckert[1], Jan Peters[1,2] and Gerhard Neumann[1]

***Abstract*— Physical interaction in robotics is a complex problem that requires not only accurate reproduction of the kinematic trajectories but also of the forces and torques exhibited during the movement. We base our approach on Movement Primitives (MP), as MPs provide a framework for modelling complex movements and introduce useful operations on the movements, such as generalization to novel situations, time scaling, and others. Usually, MPs are trained with imitation learning, where an expert demonstrates the trajectories. However, MPs used in physical interaction either require additional learning approaches, e.g., reinforcement learning, or are based on handcrafted solutions. Our goal is to learn and generate movements for physical interaction that are learned with imitation learning, from a small set of demonstrated trajectories. The Probabilistic Movement Primitives (ProMPs) framework is a recent MP approach that introduces beneficial properties, such as combination and blending of MPs, and represents the correlations present in the movement. The ProMPs provides a variable stiffness controller that reproduces the movement but it requires a dynamics model of the system. Learning such a model is not a trivial task, and, therefore, we introduce the model-free ProMPs, that are learning jointly the movement and the necessary actions from a few demonstrations. We derive a variable stiffness controller analytically. We further extent the ProMPs to include force and torque signals, necessary for physical interaction. We evaluate our approach in simulated and real robot tasks.**

I. Introduction

Developing robots that can operate in the same environment with humans and physically interacting with every-day objects requires accurate control of the contact forces that occur during the interaction. While non-compliant robots can achieve a great accuracy, the uncertainty of complex and less-structured environment prohibits physical interaction. In this paper, we focus on providing a compliant control scheme that can enable robots to manipulate their environment without damaging it. Typically, force-control requires an accurate dynamics model of the robot and its environment that is not easy to obtain. Other approaches suggest to learn a dynamics model, however, this process can be time-consuming and is prone to model-errors. We present an approach that can jointly learn the desired movement of the robot and the contact forces by human demonstrations, without relying on a learned forward or inverse model.

*The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007–2013) under grant agreements #600716 (CoDyCo) and #270327 (CompLACS)

[1]Intelligent Autonomous Systems, TU Darmstadt, 64289 Darmstadt, Germany {paraschos,neumann,rueckert}@ias.tu-darmstadt.de

[2] Robot Learning Group, Max Planck Institute for Intelligent Systems, Germany mail@jan-peters.net

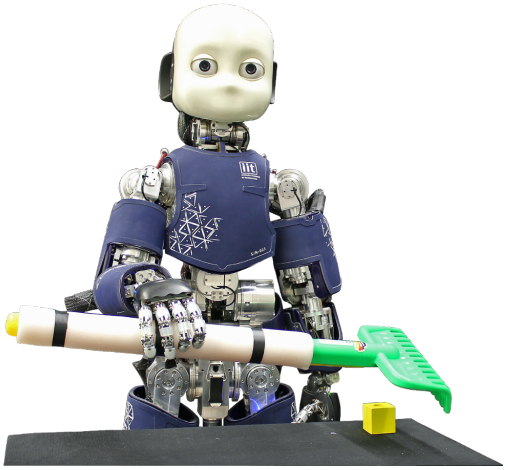

Fig. 1. The *iCub* robot is taught by imitation how to tilt a grate that we use of during the experimental evaluation of our approach. We demonstrated how to lift a grate from three different positions. Grasping from different positions change the dynamics of the task. Our method provides online adaptation and generalizes in the area of the grasps

Existing approaches for motor skill learning that are based on movement primitives [1], [2], [3], [4], [5], often incorporate into the movement primitive representation the forces needed for the physical interactions [6], [7], [8]. However, such approaches model a single successful reproduction of the task. Multiple demonstrations are typically averaged, despite that they actually represent similar, but different, solutions of the task. Thus, the applied contact forces are not correlated with the state of the robot nor sensory values that indicate the state of the environment, e.g. how heavy an object is.

In this paper, we propose learning the coordination of the interaction forces, with the kinematic state of the system, as well as the control actions needed to reproduce the movement exclusively from demonstration. Motor skill learning for such interaction tasks for high-dimensional redundant robots is challenging. This task requires real-time feedback control laws that process sensory data including joint encoders, tactile feedback and force-torque readings. We present a model-free version of the Probabilistic Movement Primitives (ProMPs) [9] that enables robots to acquire complex motor skills from demonstrations, while it can coordinate the movement with force, torque, or tactile sensing. The ProMPs have several beneficial properties, such us generalization to novel situations, combination of primitives and time-scaling, which we inherit in our approach.

ProMPs assume a locally linearizable dynamics models to

compute time-varying feedback control laws. However, such dynamics models are hard to obtain for physical interaction tasks. Therefore we obtain a time varying feedback controller directly from the demonstration without requiring such a model. In the model-free extension of the ProMPs, we condition the joint distribution over states and controls on the current state of the system, and obtain a distribution over the controls. We show that this distribution represents a time-varying stochastic linear feedback controller. Due to the time-varying feedback gains, the controller can exhibit behavior with variable stiffness and, thus, it is safe to use in physical interaction. A similar control approach has recently been presented in [10].

Our approach inherits many beneficial properties of the original ProMP formulation. We can reproduce the variability in the demonstrations and use probabilistic operators for generalization to new tasks or the co-activation of learned primitives. The resulting feedback controller shows similar properties as in the model-based ProMP approach, it can reproduce optimal behavior for stochastic systems and exactly follow the learned trajectory distribution, at least, if the real system dynamics are approximately linear for each time step. For non-linear systems, the estimated variable stiffness controller can get unstable if the robot reaches configurations that are far away from the set of demonstrations. To avoid this problem, we smoothly switch between a stable PD-controller and the ProMP controller if the support of the learned distribution for the current situation is small. We show that this extension allows us to track trajectory distributions accurately even for non-linear systems.

The model-free ProMP approach is evaluated in simulation with linear and non-linear dynamical systems in Section V. In a real task, we tilt a grate which is grasped at different positions. We show that the model-free ProMPs can generalize to different grasping locations on a grate through exploiting the correlations between motor commands and force feedback.

II. Related Work

In this section, we review related work on movement primitives for imitation learning that combine position and force tracking, model the coupling between kinematics and forces and are able to capture the correlations between these two quantities.

The benefit of an additional feedback controller to track desired reference forces was demonstrated in grasping tasks in [6]. Individual dynamical systems (DMPs) [5] were trained for both, position and force profiles in imitation learning. The force feedback controller substantially improved the success rate of grasps in tracking demonstrated contact forces under changing conditions. For manipulation tasks like opening a door, the authors showed that the learned force profiles can be further improved through reinforcement learning [7].

For many tasks, such as like bi-manual manipulations, the feedback controller needs to be coupled. Gams et al. [11] proposed *cooperative* dynamical systems, where deviations from desired forces modulate the velocity forcing term in the DMPs for position control. This approach was tested on two independently operating robot arms solving cooperative tasks like lifting a stick [8]. Deviations in the sensed contact forces in one robot were used to adapt the DMP of the other robot and the coupling parameters were obtained through iterative learning control. A related probabilistic imitation learning approach to capture the couplings in time was proposed in [12]. In this approach, Gaussian mixture models were used to represent the variance of the demonstrations. The approach was evaluated successfully on complex physical interaction tasks such as ironing, opening a door, or pushing against a wall.

Adapting Gaussian Mixture Models (GMMs) [13], [14], [15], [16] have been proposed for use in physical interaction tasks. The major difference to the dynamical systems approach is that GMMs can represent the variance of the movement. Closely related to our approach, Evrard et al. in [17] used GMMs to learn joint distributions of positions and forces. Joint distributions capture the correlations between positions and forces and were used to improve adaptation to perturbations in cooperative human robot tasks for object lifting. In this approach, the control gains were fixed to track the mean of the demonstrated trajectories. In [18], it was shown that by assuming known forward dynamics, variable stiffness control gains can be derived in closed form to match the demonstrations. We address here an important related question of how these gains can be learned in a model-free approach from the demonstrations.

III. Model-Free Probabilistic Movement Primitives

We propose a novel framework for robot control which can be employed in physical interaction scenarios. In our approach, we jointly learn the desired trajectory distribution of the robot's joints or end-effectors and the corresponding controls signals. We train our approach from a limited set of demonstrations. We refer to the joint distribution as state-action distribution. Further, we incorporate proprioceptive sensing, such as force or tactile sensing, into our state representation. The additional sensing capabilities are of high importance for physical interaction as they can disambiguate kinetically similar states. We present our approach by, first, extending the Probabilistic Movement Primitives (ProMPs) framework [9] to encode the state-action distribution and, second, we derive a stochastic feedback controller without the use of a given system dynamics model. Finally, we extend our control approach for states which are relatively far from the vicinity of the learned state-action distribution. In that case, our control approach can no longer produce correcting actions, and an additional backup controller with high gains is needed. Our framework inherits most of the beneficial properties introduced by the ProMPs that significantly improved generalization to novel situations and enables the generation of primitives that *concurrently* solve multiple tasks [9].

A. Encoding the Time-Varying State-Action Distribution of the Movement

We avoid explicitly learning robot and environment models by learning directly the appropriate control inputs, while keeping the beneficial properties of the ProMP approach, such us generalization and concurrent execution.

In order to simplify the illustration of our approach, we first discuss the special case of a single Degree of Freedom (DoF) and, subsequently, we expand our description to the generic case of multiple DoF. The description is based in [9], but modified appropriately to clarify how the actions can be modelled. First, we define the extended state of the system as

$$\boldsymbol{y}_t = [q_t, \dot{q}_t, u_t]^T, \tag{1}$$

where q_t is the position of the joint, $\dot{q}_t$ the velocity, and u_t the control applied at time-step t. Similar to ProMPs, we use a linear basis function model to encode the trajectory of the extended state $\boldsymbol{y}_t$. The feature matrix and the weight vector of the non-linear function approximation model become

$$\boldsymbol{y}_t = \begin{bmatrix} q_t \\ \dot{q}_t \\ u_t \end{bmatrix} = \tilde{\boldsymbol{\Phi}}_t \boldsymbol{w}, \quad \tilde{\boldsymbol{\Phi}}_t = \begin{bmatrix} \boldsymbol{\phi}_t^T & \mathbf{0} \\ \dot{\boldsymbol{\phi}}_t^T & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\psi}_t^T \end{bmatrix}, \boldsymbol{w} = \begin{bmatrix} \boldsymbol{w}_q \\ \boldsymbol{w}_u \end{bmatrix}, \tag{2}$$

where the vectors $\boldsymbol{\phi}_t$ and $\boldsymbol{\psi}_t$ represent the feature vectors for the position q_t and the control u_t respectively. The derivative of the position feature vector $\dot{\boldsymbol{\phi}}_t$ is used to compute the velocity of the joint $\dot{q}_t$. The weight vector $\boldsymbol{w}$ contains the weight vector for the position $\boldsymbol{w}_q$ and the weight vector for the control $\boldsymbol{w}_u$. The dimensionality of the feature $\boldsymbol{\phi}_t$ and weight $\boldsymbol{w}_q$ vectors is $N \times 1$, where N is the number of features used to encode the joint position. Similarly, the dimensionality of $\boldsymbol{\psi}_t$ and $\boldsymbol{w}_u$ vectors is $M \times 1$. The remaining entries of $\tilde{\boldsymbol{\Phi}}_t$, denoted by $\mathbf{0}$, are zero-matrices with the appropriate dimensionality. In our approach, we distinct between the features used to encode the position from the features used to encode the control signal due to the different properties of the two signals. The distinction allows us to use of different type of basis functions, different parameters, or a different number of basis functions.

We extend our description to the multidimensional case. First, we extend the state of the system from Equation (2) to

$$\boldsymbol{y}_t = \left[\boldsymbol{q}_t^T, \dot{\boldsymbol{q}}_t^T, \boldsymbol{u}_t^T\right]^T, \tag{3}$$

where the vector $\boldsymbol{q}_t$ is a concatenation of the positions of all joints of the robot, the vector $\dot{\boldsymbol{q}}_t$ of the velocities of the joints, and $\boldsymbol{u}_t$ of the controls respectively. The feature matrix $\tilde{\boldsymbol{\Phi}}_t$ now becomes a block matrix

$$\tilde{\boldsymbol{\Phi}}_t = \left[\boldsymbol{\Phi}_t^T, \dot{\boldsymbol{\Phi}}_t^T, \boldsymbol{\Psi}_t^T\right]^T, \tag{4}$$

where

$$\boldsymbol{\Phi}_t = \left[\begin{array}{ccc|c} \boldsymbol{\phi}_t^T & \cdots & \mathbf{0} & \\ \vdots & \ddots & \vdots & \mathbf{0} \\ \mathbf{0} & \cdots & \boldsymbol{\phi}_t^T & \end{array}\right], \tag{5}$$

$$\boldsymbol{\Psi}_t = \left[\begin{array}{c|ccc} & \boldsymbol{\psi}_t^T & \cdots & \mathbf{0} \\ \mathbf{0} & \vdots & \ddots & \vdots \\ & \mathbf{0} & \cdots & \boldsymbol{\psi}_t^T \end{array}\right], \tag{6}$$

define the features for the joint positions and the joint controls. Similarly to the single DoF, the features used for the joint velocities $\dot{\boldsymbol{\Phi}}_t$ are the time derivatives of the features of the joint positions $\boldsymbol{\Phi}_t$. We use the same features for every DoF. The dimensionality of the feature matrices $\boldsymbol{\Phi}_t$ and $\boldsymbol{\Psi}_t$ is $K \times K \cdot (N + M)$, where K denotes the number of DoF.

The weight vector $\boldsymbol{w}$ has a similar structure to Equation (2) and, for the multi-DoF case, is given by

$$\boldsymbol{w} = \Big[\underbrace{{}^1\boldsymbol{w}_q^T, \cdots, {}^K\boldsymbol{w}_q^T}_{\text{weights for joint positions}}, \underbrace{{}^1\boldsymbol{w}_u^T, \cdots, {}^K\boldsymbol{w}_u^T}_{\text{weights for joint controls}} \Big]^T, \tag{7}$$

where ${}^i\boldsymbol{w}$ denotes the weight vector for joint $i \in [1, K]$.

The probability of a single trajectory $\boldsymbol{\tau} = \{\boldsymbol{y}_t, t \in [1 \cdots T]\}$, composed from states of T subsequent time steps, given the parameters $\boldsymbol{w}$, is computed by

$$p(\boldsymbol{\tau}|\boldsymbol{w}) = \prod_t \mathcal{N}\left(\boldsymbol{y}_t | \boldsymbol{\Phi}_t \boldsymbol{w}, \boldsymbol{\Sigma}_{\boldsymbol{y}}\right), \tag{8}$$

where we assume *i.i.d.* Gaussian observation noise with zero mean and $\boldsymbol{\Sigma}_{\boldsymbol{y}}$ covariance. Representing multiple trajectories would require a set of weights $\{\boldsymbol{w}\}$. Instead of explicitly maintaining such a set, we introduce a distribution over the weights $p(\boldsymbol{w}; \boldsymbol{\theta})$, where the parameter vector $\boldsymbol{\theta}$ defines the parameters of the distribution. Given the distribution parameters $\boldsymbol{\theta}$, the probability of the trajectory becomes

$$p(\boldsymbol{\tau}; \boldsymbol{\theta}) = \int p(\boldsymbol{\tau}|\boldsymbol{w}) p(\boldsymbol{w}; \boldsymbol{\theta}) \mathrm{d}\boldsymbol{w}, \tag{9}$$

where we marginalize over the weights $\boldsymbol{w}$. As in the ProMP approach, we use a Gaussian distribution to represent $p(\boldsymbol{w}; \boldsymbol{\theta})$, where $\boldsymbol{\theta} = \{\boldsymbol{\mu}_{\boldsymbol{w}}, \boldsymbol{\Sigma}_{\boldsymbol{w}}\}$. Using a Gaussian distribution enables the marginal to be computed analytically and facilitates learning. The distribution over the weight vector $p(\boldsymbol{w}; \boldsymbol{\theta})$ correlates (couples) the DoFs of the robot to the action vector at every time-step t. The probability of the current state-action vector $\boldsymbol{y}_t$ given $\boldsymbol{\theta}$ is computed by

$$\begin{aligned} p(\boldsymbol{y}_t; \boldsymbol{\theta}) &= \int \mathcal{N}\left(\boldsymbol{y}_t | \boldsymbol{\Phi}_t \boldsymbol{w}, \boldsymbol{\Sigma}_{\boldsymbol{y}}\right) \mathcal{N}\left(\boldsymbol{w} | \boldsymbol{\mu}_{\boldsymbol{w}}, \boldsymbol{\Sigma}_{\boldsymbol{w}}\right) \mathrm{d}\boldsymbol{w} \\ &= \mathcal{N}\left(\boldsymbol{y}_t \middle| \boldsymbol{\Phi}_t \boldsymbol{\mu}_{\boldsymbol{w}}, \boldsymbol{\Phi}_t \boldsymbol{\Sigma}_{\boldsymbol{w}} \boldsymbol{\Phi}_t^T + \boldsymbol{\Sigma}_{\boldsymbol{y}}\right), \end{aligned} \tag{10}$$

in closed form. We use normalized Gaussian basis functions as features. Each basis function is defined in the time domain by

$$\phi_i(t) = \frac{b_i(t)}{\sum_{j=1}^n b_j(t)}, \tag{11}$$

$$b_i(t) = \exp\left(-\frac{(t - c_i)^2}{2h}\right), \tag{12}$$

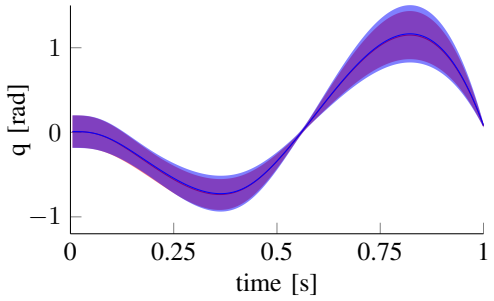

Fig. 2. We evaluate our approach on a simulated 1-DoF linear system. We use $N = 30$ demonstrations (red) for training. During the reproduction (blue) our approach matches exactly the demonstrations.

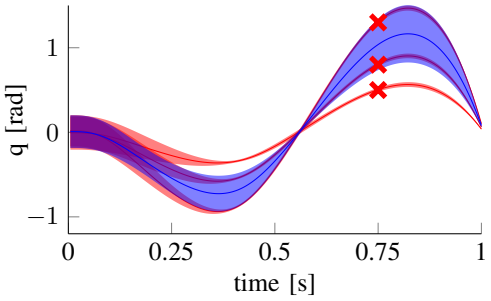

Fig. 3. We evaluate the generalization capabilities of our approach with *conditioning*. The initial distribution is depicted in blue. At time $t = 0.75s$ we condition the initial distribution to pass at a specific position $q = \{0.5, 0.8, 1.3\}$ with low variance. We generate $N = 30$ demonstrations for every conditioning point and we show the resulting distribution in red. The *X* markers denotes the position at the conditioning point.

where c_i denotes the center of the ith basis function and h the bandwidth. The centers of the basis functions are spread uniformly in $[-2h, T_{\text{end}}+2h]$. The number of basis functions and the bandwidth value we used, depend on the complexity of task. Typically, complex task require higher number of basis functions in order to represent them accurately.

B. Imitation Learning for Model-Free ProMPs

We use multiple demonstrations to estimate the parameters $\boldsymbol{\theta} = \{\boldsymbol{\mu}_{\boldsymbol{w}}, \boldsymbol{\Sigma}_{\boldsymbol{w}}\}$ of the distribution over the weights $p(\boldsymbol{w}|\boldsymbol{\theta})$. First, for each demonstration i, we use linear ridge regression to estimate the parameter vector $\boldsymbol{w}_i$ associated to that specific demonstration, i.e.,

$$\boldsymbol{w}_i = (\boldsymbol{\Phi}_t^T \boldsymbol{\Phi}_t + \lambda \boldsymbol{I})^{-1} \boldsymbol{\Phi}_t^T \boldsymbol{Y}_i, \tag{13}$$

where λ denotes the ridge factor and $\boldsymbol{Y}_i$ the observations of the state and action for all the time steps of that demonstration. We set λ to zero, unless numerical issues arise. Subsequently, we estimate the parameters $\boldsymbol{\theta}$ from the set of weights $\{\boldsymbol{w}_i, i \in [1, N]\}$ using the ML estimators for Gaussians, i.e.,

$$\begin{aligned} \boldsymbol{\mu}_{\boldsymbol{w}} &= \frac{1}{L} \sum_{i=1}^{L} \boldsymbol{w}_i, \\ \boldsymbol{\Sigma}_{\boldsymbol{w}} &= \frac{1}{L} \sum_{i=1}^{L} (\boldsymbol{w}_i - \boldsymbol{\mu}_{\boldsymbol{w}})(\boldsymbol{w}_i - \boldsymbol{\mu}_{\boldsymbol{w}})^T, \end{aligned} \tag{14}$$

where L is the number of demonstrations.

C. Integration of Proprioceptive Feedback

Additional sensory feedback integration, e.g., force-torque feedback, is beneficial for physical interaction scenarios as we can capture the correlation of the trajectory, the controls and the sensory signal. This correlation might contain useful information for the reproduction of the movement. We extend our approach to additionally contain the sensory signal $\boldsymbol{s}_t$. The extension require the state $\boldsymbol{y}_t$ to include the sensory signal $\boldsymbol{s}_t$. We estimate an individual weight vector $\boldsymbol{w}_s$ that we include in the concatenated weight vector $\boldsymbol{w}$. Hence, by learning the distribution $p(\boldsymbol{w})$, we can represent the correlations between the sensory signal and the control commands. We use the sensory signal to get a new desired trajectory distribution and its controls.

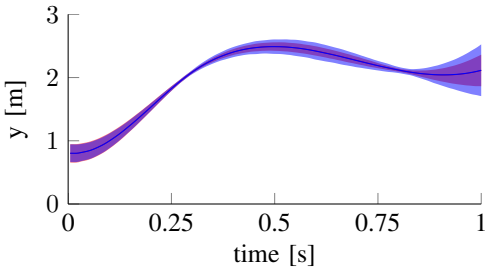

Fig. 4. We evaluate our approach on an non-linear system with $D = 4$ DoF. While the dynamics of the task are non-linear we are able to reproduce (blue) accurately the demonstrated distribution (red). We show the trajectory distribution of the "y" dimension of the task-space of the robot. Our approach captures the correlations between the DoF of the robot and reduces the variance of the trajectory reproduction at both via-points.

D. Generalization with Conditioning

The modulation of via-points and final positions is an important property of any MP framework to adapt to new situations. Generalization to different via-points or final targets can be implemented by conditioning the distribution at reaching the desired position $\boldsymbol{q}_t^*$ (or by conditioning on any other sensory value) at time step t.

By applying Bayes theorem, we obtain a new distribution $p(\boldsymbol{w}|\boldsymbol{q}_t^*)$ for $\boldsymbol{w}$ which is Gaussian with mean and variance

$$\boldsymbol{\mu}_{\boldsymbol{w}}^{[\text{new}]} = \boldsymbol{\mu}_{\boldsymbol{w}} + \boldsymbol{Q}_t \left(\boldsymbol{q}_t^* - \boldsymbol{\Psi}_t^T \boldsymbol{\mu}_{\boldsymbol{w}} \right), \tag{15}$$

$$\boldsymbol{\Sigma}_{\boldsymbol{w}}^{[\text{new}]} = \boldsymbol{\Sigma}_{\boldsymbol{w}} - \boldsymbol{Q}_t \boldsymbol{\Psi}_t^T \boldsymbol{\Sigma}_{\boldsymbol{w}}, \tag{16}$$

$$\boldsymbol{Q}_t = \boldsymbol{\Sigma}_{\boldsymbol{w}} \boldsymbol{\Psi}_t \left(\boldsymbol{\Sigma}_q^* + \boldsymbol{\Psi}_t^T \boldsymbol{\Sigma}_{\boldsymbol{w}} \boldsymbol{\Psi}_t \right)^{-1}, \tag{17}$$

where $\boldsymbol{\Sigma}_q^*$ is a covariance matrix specifying the accuracy of the conditioning. As the weight vectors for the controls are also contained in the distribution, the distribution over the controls will change accordingly, such that, by executing the controls, we will reach the desired state $\boldsymbol{q}_t^*$.

E. Robot Control with Model-Free ProMPs

We derive a stochastic feedback controller which is ideally capable of reproducing the learned distribution. We define as $\tilde{\boldsymbol{y}}_t$ the observable state of the system, that contains the joint

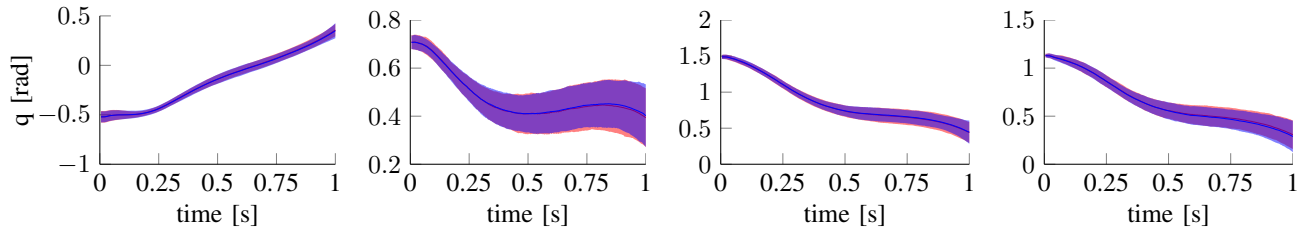

Fig. 5. The evaluation of our approach on the quad-link robot. We present the results of the of the DoF in joint space. The demonstrated distribution is plotted in red and the reproduction in blue. The two distributions match. The two via-points of the movement, which were set in task-space, are not visible in joint-space.

positions, velocities, and potentially force or torque data, but not the action. We rewrite the joint probability

$$p(\boldsymbol{y}_t) = p(\tilde{\boldsymbol{y}}_t, \boldsymbol{u}_t) = \mathcal{N}\left(\begin{bmatrix}\tilde{\boldsymbol{y}}_t \\ \boldsymbol{u}_t\end{bmatrix} \middle| \tilde{\boldsymbol{\Phi}}_t \boldsymbol{\mu}_{\boldsymbol{w}}, \tilde{\boldsymbol{\Phi}}_t \boldsymbol{\Sigma}_{\boldsymbol{w}} \tilde{\boldsymbol{\Phi}}_t^T + \boldsymbol{\Sigma}_{\boldsymbol{y}}\right), \quad (18)$$

where

$$\tilde{\boldsymbol{\Phi}}_t \boldsymbol{\Sigma}_{\boldsymbol{w}} \tilde{\boldsymbol{\Phi}}_t^T = \begin{bmatrix}\boldsymbol{\Phi}_t \boldsymbol{\Sigma}_{\boldsymbol{w}} \boldsymbol{\Phi}_t^T & \boldsymbol{\Phi}_t \boldsymbol{\Sigma}_{\boldsymbol{w}} \boldsymbol{\Psi}_t^T \\ \boldsymbol{\Psi}_t \boldsymbol{\Sigma}_{\boldsymbol{w}} \boldsymbol{\Phi}_t^T & \boldsymbol{\Psi}_t \boldsymbol{\Sigma}_{\boldsymbol{w}} \boldsymbol{\Psi}_t^T\end{bmatrix}, \quad (19)$$

and condition on the current observable state $\tilde{\boldsymbol{y}}_t$ to obtain the desired action. From the Bayes theorem, we obtain the probability of the desired action

$$p(\boldsymbol{u}_t | \tilde{\boldsymbol{y}}_t) = \frac{p(\tilde{\boldsymbol{y}}_t, \boldsymbol{u}_t)}{p(\tilde{\boldsymbol{y}}_t)} = \mathcal{N}\left(\boldsymbol{u}_t | \boldsymbol{\mu}_{\boldsymbol{u}}, \boldsymbol{\Sigma}_{\boldsymbol{u}}\right), \quad (20)$$

which is a Gaussian distribution as both $p(\tilde{\boldsymbol{y}}_t)$ and $p(\boldsymbol{u}_t)$ are Gaussian. The mean and covariance of $p(\boldsymbol{u}_t)$ are computed by

$$\boldsymbol{\mu}_{\boldsymbol{u}} = \boldsymbol{\Psi}_t \boldsymbol{\mu}_{\boldsymbol{w}} + \boldsymbol{K}_t, (\tilde{\boldsymbol{y}}_t - \boldsymbol{\Phi}_t \boldsymbol{\mu}_{\boldsymbol{w}}) \quad (21)$$

$$\boldsymbol{\Sigma}_{\boldsymbol{u}} = \boldsymbol{\Psi}_t \boldsymbol{\Sigma}_{\boldsymbol{w}} \boldsymbol{\Psi}_t^T + \boldsymbol{K}_t \boldsymbol{\Phi}_t \boldsymbol{\Sigma}_{\boldsymbol{w}} \boldsymbol{\Phi}_t^T, \quad (22)$$

$$\boldsymbol{K}_t = \boldsymbol{\Psi}_t \boldsymbol{\Sigma}_{\boldsymbol{w}} \boldsymbol{\Phi}_t^T \left(\boldsymbol{\Phi}_t \boldsymbol{\Sigma}_{\boldsymbol{w}} \boldsymbol{\Phi}_t^T\right)^{-1}, \quad (23)$$

using Gaussian identities. We rewrite the mean control given the observable state $\tilde{\boldsymbol{y}}_t$ as

$$\begin{aligned}\boldsymbol{\mu}_{\boldsymbol{u}} &= \boldsymbol{\Psi}_t \boldsymbol{\mu}_{\boldsymbol{w}} + \boldsymbol{K}_t \tilde{\boldsymbol{y}}_t - \boldsymbol{K}_t \boldsymbol{\Phi}_t \boldsymbol{\mu}_{\boldsymbol{w}} \\ &= \boldsymbol{K}_t \tilde{\boldsymbol{y}}_t + \boldsymbol{k}_t, \end{aligned} \quad (24)$$

and observe that it has the same structure as a feedback controller with time varying gains. The feedback gain matrix $\boldsymbol{K}_t$ couples the DoF and the additional force-torque signals of the system. The control covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{u}}$ introduces correlated noise in the controls. The noise used only if we want to match the variability of the demonstrations. Alternatively, we can disable the noise and replay the noise-free behavior.

F. Correction Terms for Non-Linear Systems

A basic assumption for the linear feedback controller obtained by the ProMP approach is that the movement is defined in a local vicinity such that a linear controller is sufficient. Whenever the robot's state "leaves" this vicinity, due to the non-linearities of the dynamics, the learned

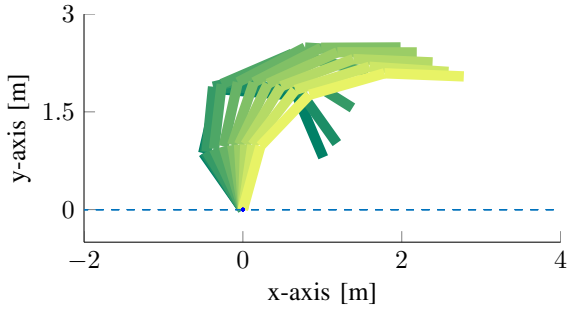

Fig. 6. An animation of the movement of the quad-link non-linear robot during the execution of our approach. We use darker colors at the beginning of the movement and lighter at the end.

feedback controller might not be able to direct the robot back to the desired trajectory distribution. Therefore, we apply a correction controller that is active only when the state is sufficiently "far" outside the distribution and directs the system to the mean of the demonstrated state distribution. The correction controller is defined as a standard PD controller with hand-tuned gains, i.e.,

$$\boldsymbol{u}_t^C = \boldsymbol{K}_P \left(\boldsymbol{\mu}_{q,t} - \boldsymbol{q}_t\right) + \boldsymbol{K}_D \left(\boldsymbol{\mu}_{\dot{q},t} - \dot{\boldsymbol{q}}\right) + \boldsymbol{u}_{\text{ff},t}, \quad (25)$$

where the feed forward term $\boldsymbol{u}_{\text{ff},t}$ is still estimated from the ProMP and given by the mean action of the ProMP for time step t, i.e.,

$$\boldsymbol{u}_{\text{ff},t} = \boldsymbol{K}_t \boldsymbol{\Phi}_t \boldsymbol{\mu}_{\boldsymbol{w}} + \boldsymbol{k}_t. \quad (26)$$

The correcting action $\boldsymbol{u}_t^C$ is only applied if we are outside the given trajectory distribution. We use a sigmoid activation function that depends on the log-likelihood of the current state to switch between the ProMP feedback controller and the correction controller,

$$\sigma\left(\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t\right) = \frac{1}{1 + \exp\left(-\log\left(p\left(\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t; \boldsymbol{\theta}\right)\right) \beta^{-1} - \alpha\right)}, \quad (27)$$

where α and β are hand tuned parameters of the activation function. We linearly interpolate between the controls of the ProMP and the correction action. For a high likelihood, e.g., $\sigma(\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t) = 1$ we fully activate the feedback controller from the ProMP. For $\sigma(\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t) = 0$ we fully activate the correction action.

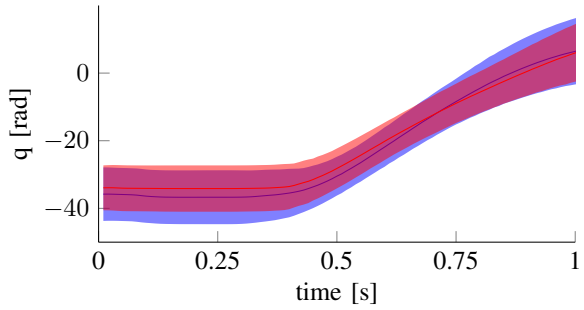

Fig. 7. The trajectory distribution of the wrist joint of the iCub during our experiment. The demonstrated distribution is presented in blue and the reproduction in red. The demonstrated distribution contain trajectories from all three grasping locations. The reproduction distribution contain trajectories from seven grasping locations. The Model-Free ProMPs can reproduce the demonstrated distribution in new grasping locations.

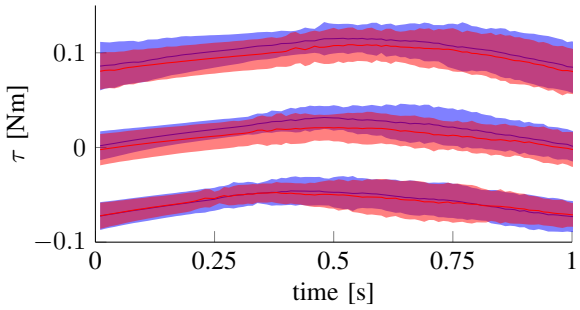

Fig. 8. The torque distribution of all grasping locations used during the demonstrations. Each location created a distinct offset in the measured torque. We present the demonstrated torque distributions in blue. Additionally, we show that our approach can reproduce the torque distribution when we position the grate at the same locations as in the demonstrations. We present the reproduction results in red.

IV. Experimental Evaluation

We begin our experimental evaluation with different toy tasks to demonstrate the properties of the model-free ProMP approach. First, we demonstrate that our model-free ProMP controller can reproduce the demonstrated trajectory distribution accurately on a linear 1-D system. Then, we change the desired trajectory distribution by conditioning, to generalize to different via-points and we execute our controller. We shot that the resulting distribution exactly reaches the via-points.

In a sub-sequent experiment, we test the model-free ProMP on a non-linear Quad-Link pendulum. We show that by the use of the correcting PD controller we can still track the distribution accurately.

Finally, we performed first experiments on the iCub, where the humanoid is grasping a grate at different grasp locations and has to tilt it. By learning the correlation between the force-torque readings and the demonstrated control actions the iCub should learn to compensate for gravitational effects.

A. Reproduction of the Trajectory Distribution

We illustrate our approach in an one dimensional linear second order integrator as the underlying dynamical system. We created the demonstrations by first creating different desired trajectories with splines that go through different via-points. The real trajectories are created by following a given spline with a PD control law. We also added noise to the acceleration of the system. The resulting trajectory distribution is given in Fig. 2 (red). In the same figure, we illustrated the resulting trajectory distribution by using the ProMP controller from the learned model-free ProMP. As we can see, the controller could match the distribution accurately.

B. Generalization by Conditioning to Different Via-Points

We test the conditioning operations that can be performed upon the trajectory distribution to generalize to different via-points. We conditioned the trajectory distribution to reach the positions $0.5, 0.8$ and 1.3 respectively at time point $t = 0.75s$. The resulting via points are indicated by a red cross in Fig. 3. For each of the conditioning scenarios, we plot the

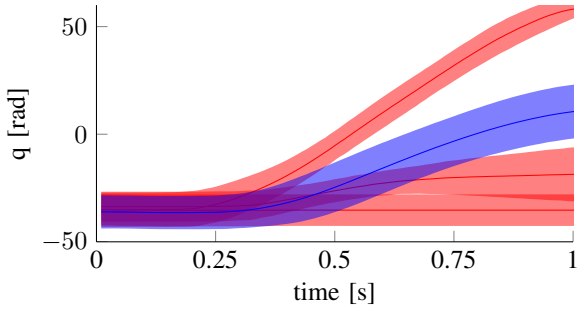

Fig. 9. The trajectory distribution of the wrist joint of the robot, when we disable the torque feedback. Depending on the grasping location, the robot either fails to lift the grate to the same height as demonstrated, or, it overshoots the lifting task due to gravity. In the later case, it should be noted that the center of mass of the grate is moved over the axis of the joint and, thus, gravity forces the grate to lift. For comparison, we present the demonstration distribution from all grasping locations in blue.

resulting trajectory distribution when executing the controller of the conditioned ProMP. The model-free ProMP controller keeps the shape of the distribution while reaching the desired via-points.

C. Non-Linear Quad-Link Pendulum

To evaluate the quality of our controller on a non-linear system, we tested our model-free ProMP approach on a non-linear quad-link planar pendulum. Each link had a mass of 1kg and a length of 1m. We used the standard rigid body dynamics equations, where the gravity and the Coriolis forces are the major non-linear terms. We collected demonstrations by defining the desired trajectory as a spline with two via-point at $t = 0.3, 0.8$ in the task-space of the robot. We generated the demonstration trajectories using inverse kinematics for generating the joint space reference trajectories. Then, we used a inverse dynamics controller to track the reference trajectories and we collected the joint state-action data. We trained our approach using $N = 30$ demonstrations.

The resulting trajectory distribution for the y-dimension of the task-space is show in Fig. 4. The robot can track with its end-effector the desired distribution accurately and can reproduce the two via-points. In Fig. 5 we show all four

the joint trajectories. In the joint space distributions the via-points are not visible but are captured in the covariance matrix of the weights. While the distribution is a wide, the controller could match the mean and variance of the demonstrated trajectory distribution. In Fig. 6, we illustrated the resulting trajectory from the controller in the task space of the robot. The activation of the correcting controller is around 1% of the total execution time.

D. Adaptation to External Forces on the ICub

In this experiment we used the presented model-free ProMP approach to learn a one-dimensional torque feedback controller in the humanoid robot iCub. The task is to tilt a grate multiple times from an initial distribution to a goal distribution, as show in Fig. 7. In our experiments we use the wrist joint. The grate is attached to the robot at different lengths, to simulate different grasping locations. We demonstrate 20 movements per grasping location to train our approach. The data where recorded through tele-operation. In this experiment the state encodes the joint angle encoder value and the joint torque reading in the wrist. We present the recored torques from the sensor of the robot for all three demonstrated grasping locations in Fig. 8. By placing the grate on the same location as during the demonstration and reproducing the movement with our approach, we show that we observe the same torque profile. The force measurement is crucial in our experiment as it is used for applying the correct forces during the execution of the movement. When disabled, the robot either fails to lift the grate to the demonstrated location or it overshoots. The overshooting is due to gravity, as in that grasping location the center of the mass of the grate is moved over the axis of wrist rotation. The results are shown in Fig. 9. The reproduction distributions where created using twenty executions of the model-free ProMP controller per grasping location. Our approach can generalize to different grasping locations between the demonstrations. We generalized into four new locations and executed our controller. The robot reproduces the same joint distribution while compensating for the different dynamics, as shown in Fig.7.

V. Conclusion

In this paper, we presented a model-free approach for Probabilistic Movement Primitives (ProMP) that can be used for learning skills for physical interaction with the environment from demonstrations. In contrast to the original approach, the model-free ProMP approach does not require a known model of the system dynamics as the stochastic feedback controller is directly obtained from the estimated distribution over the trajectories, which includes the control signals. We showed that the model-free ProMP approach inherits many beneficial properties from the original ProMPs such as reproducing the variability in the demonstrations as well as using probabilistic operations such as conditioning for generalization to different via points. Our approach is different from directly encoding the actions, as generates the action through a model that depends on the state and the time. Hence, our approach can generalize well in the vicinity of the demonstrations. Our approach can be used in tasks where time is critical for the execution of the task, e.g. pushing a button at a specific movement, or grasping a moving object.

For learning physical interaction tasks, we showed that we can include sensory signals, for example the measure torques, in our distribution. By learning the correlations of this sensory signal, we can coordinate the controls needed for the physical interaction with the measured torques and forces. Such coordination is essential for the complex interaction tasks. In a preliminary study, we showed how the model-free ProMP approach can be applied to the iCub to apply forces to objects with unknown masses. In future work, we will investigate the use of model-free ProMPs for more complex scenarios.

References

[1] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Learning attractor landscapes for learning motor primitives," in *Advances in Neural Information Processing Systems (NIPS)*, 2003.

[2] S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert, "Learning movement primitives," in *Int. Symp. on Robotics Research.*, 2005.

[3] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, "Robot programming by demonstration," in *Springer handbook of robotics*, 2008.

[4] J. Kober and J. Peters, "Learning motor primitives for robotics," in *Int. Conf. on Robotics and Automation (ICRA)*, 2009.

[5] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: learning attractor models for motor behaviors," *Neural Computation*, 2013.

[6] P. Pastor, L. Righetti, M. Kalakrishnan, and S. Schaal, "Online movement adaptation based on previous sensor experiences," in *Int. Conf. on Intelligent Robots and Systems (IROS)*, 2011.

[7] M. Kalakrishnan, L. Righetti, P. Pastor, and S. Schaal, "Learning force control policies for compliant manipulation," in *Int. Conf. on Intelligent Robots and Systems (IROS)*, 2011.

[8] A. Gams, B. Nemec, A. J. Ijspeert, and A. Ude, "Coupling movement primitives: Interaction with the environment and bimanual tasks," *IEEE Transactions on Robotics*, 2014.

[9] A. Paraschos, C. Daniel, J. Peters, and G. Neumann, "Probabilistic movement primitives," in *Advances in Neural Information Processing Systems (NIPS)*, 2013.

[10] A. X. Lee, H. Lu, A. Gupta, S. Levine, and P. Abbeel, "Learning force-based manipulation of deformable objects from multiple demonstrations," in *Int. Conf. on Robotics and Automation (ICRA)*, 2015.

[11] A. Gams, M. Do, A. Ude, T. Asfour, and R. Dillmann, "On-line periodic movement and force-profile learning for adaptation to new surfaces," in *Int. Conf. on Humanoid Robots (Humanoids)*, 2010.

[12] P. Kormushev, D. N. Nenchev, S. Calinon, and D. G. Caldwell, "Upper-body kinesthetic teaching of a free-standing humanoid robot," in *Int. Conf. on Robotics and Automation (ICRA)*, 2011.

[13] S. Calinon, F. D'halluin, E. L. Sauser, D. G. Caldwell, and A. G. Billard, "Learning and reproduction of gestures by imitation," *IEEE Robotics and Automation Magazine*, Jun. 2010.

[14] P. Kormushev, S. Calinon, and D. G. Caldwell, "Imitation learning of positional and force skills demonstrated via kinesthetic teaching and haptic input," *Advanced Robotics*, 2011.

[15] K. Kronander and A. Billard, "Learning compliant manipulation through kinesthetic and tactile human-robot interaction," *IEEE Transactions on Haptics*, 2013.

[16] S. Calinon, D. Bruno, and D. G. Caldwell, "A task-parameterized probabilistic model with minimal intervention control," in *Int. Conf. on Robotics and Automation (ICRA)*, 2014.

[17] P. Evrard, E. Gribovskaya, S. Calinon, A. Billard, and A. Kheddar, "Teaching physical collaborative tasks: object-lifting case study with a humanoid," in *Int. Conf. on Humanoid Robots (Humanoids)*, 2009.

[18] E. Gribovskaya, A. Kheddar, and A. Billard, "Motion learning and adaptive impedance for robot control during physical interaction with humans," in *Int. Conf. on Robotics and Automation (ICRA)*, 2011.

Chapter 4

Robust Policy Updates for Stochastic Optimal Control (TUD)

Robust Policy Updates for Stochastic Optimal Control

Elmar Rueckert[1], Max Mindt[1], Jan Peters[1,2] and Gerhard Neumann[3]

***Abstract*— For controlling high-dimensional robots, most stochastic optimal control algorithms use approximations of the system dynamics and of the cost function (e.g., using linearizations and Taylor expansions). These approximations are typically only locally correct, which might cause instabilities in the greedy policy updates, lead to oscillations or the algorithms diverge. To overcome these drawbacks, we add a regularization term to the cost function that punishes large policy update steps in the trajectory optimization procedure. We applied this concept to the Approximate Inference Control method (AICO), where the resulting algorithm guarantees convergence for uninformative initial solutions without complex hand-tuning of learning rates. We evaluated our new algorithm on two simulated robotic platforms. A robot arm with five joints was used for reaching multiple targets while keeping the roll angle constant. On the humanoid robot Nao, we show how complex skills like reaching and balancing can be inferred from desired center of gravity or end effector coordinates.**

I. INTRODUCTION

Typical whole body motor control tasks of humanoids, like reaching for objects while walking, avoiding obstacles during motion, or maintaining balance during movement execution, can be characterized as optimization problems with multiple criteria of optimality or objectives. The objectives may be specified in the robot's configuration space (e.g., joint angles, joint velocities and base reference frame), in task space (where objectives such as desired end effector coordinates or center of gravity positions are specified), or in combinations of both. In this paper, we consider control problems in nonlinear systems with multiple objectives in combinations of these spaces.

A common strategy to whole body motor control is to separate the redundant robot's configuration space into a task space and an orthogonal null space. Objectives or optimality criteria of motion are implemented as weights or priorities [1] to the redundant solutions in the null space. While these approaches have been successfully applied to a variety of tasks, including reaching, obstacle avoidance, walking and maintaining stability [2]–[5], the application of these methods is typically limited to motor control and can not be directly used for motor planning. It is also unclear how these methods can be applied to motor control problems in nonlinear systems like compliant robots.

Alternatively, in Stochastic Optimal Control (SOC) problems [6], a movement policy is optimized with respect to a cost function, which combines the different criteria of optimality with different weightings. For nonlinear systems, SOC methods use approximations of the system dynamics and of the cost functions, e.g., through linearizations and 2nd order Taylor expansions. These approximations are only locally correct and the updates of the policy may become unstable if the minima is not close to the points of the linearizations, or may oscillate in the case of multiple solutions.

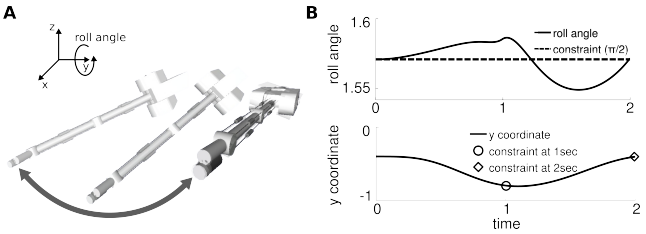

Fig. 1. A 5-degree-of-freedom robot arm has to reach for a via-point (the posture on the left in A) and return to its initial pose (the posture on the right in A). The reaching task is encoded in four task objectives, i.e., three Cartesian coordinates and the roll angle of the end effector. The inferred trajectories for the y coordinate and the roll angle, including the objectives, are shown in (B).

Many SOC methods address this issue and implement regularizations on the *algorithmic level*. E.g., in the *iLQG* method [7] a diagonal regularization term is added to the control cost Hessian[1], and in an extension [8], it was suggested to penalize deviations from the state trajectory used for linearization rather than controls. A drawback of this approach is that the additive regularization term needs rapid re-scaling to prevent divergence and accurate fine-tuning of a learning rate to find good solutions, which is challenging and increases the computational time of the algorithm.

Probabilistic planning methods that translate the SOC problem into an inference problem, typically implement learning rates in their belief updates [9] or in the feedback controller [10]. However, in nonlinear systems, both strategies are suboptimal in the sense that even with a small learning rate on the beliefs the corresponding control updates might be large (and vice-versa, respectively).

We propose to regularize the policy updated on the *cost function level* for probabilistic planning. We also penalize large distances between two successive trajectories in the iterative trajectory optimization procedure. In [8], the regularization term is only used for the control gains and not for the updates of the value function. However, the deviation from the linearization point can still be high if small regularization

[1]Intelligent Autonomous Systems Lab, Technische Universität Darmstadt, Hochschulstr. 10, 64289 Darmstadt, Germany {rueckert, mindt}@ias.tu-darmstadt.de

[2]Robot Learning Group, Max-Planck Institute for Intelligent Systems, Tuebingen, Germany mail@jan-peters.net

[3]Computational Learning for Autonomous Systems, Hochschulstr. 10, 64289 Darmstadt, Germany neumann@ias.tu-darmstadt.de

[1]The update step in the trajectory optimizer corresponds to a Gauss-Newton Hessian approximation [8].

terms are used. In our approach, we always want to stay close to the linearization point as the used approximations are only locally correct. Hence, using too large update steps by greedily exploiting the inaccurate models might again be dangerous, leading the instabilities or oscillations. The scaling parameter of our punishment term serves as step size of the policy update. Due to the use of probabilistic planning, the need of an additional learning rate and complex update strategies of this learning rate can be avoided. Moreover, we will demonstrate that this type of regularization results in more robust policy updates in comparison to [8]. We choose the Approximate Inference Control (AICO) algorithm as probabilistic planning method [9] to discuss and analyze the proposed regularization, however, the same "*trick*" can be applied to large variety of SOC methods.

In the rest of this paper, we introduce the probabilistic planning method AICO, analyze its convergence properties in a reaching task in a light-weight robot arm and introduce the proposed regularization on the *cost function level*. The resulting algorithm is evaluated on the humanoid robot Nao, where in first results, arm reaching and balancing skill are inferred from desired center of gravity or end effector coordinates. We conclude in Section IV.

II. METHODS

We consider finite horizon Markov decision problems[2]. Let $\boldsymbol{q}_t \in \mathbb{Q}$ denote the current robot's state in configuration space (e.g., a concatenation of joint angles, joint velocities and reference coordinates in floating base systems) and let vector $\boldsymbol{x}_t \in \mathbb{X}$ denote task space features like end effector positions or the center of gravity of a humanoid (these features will be used to specify a cost function later). At time t, the robot executes the action $\boldsymbol{u}_t \in \mathbb{U}$ according to the movement policy $\pi(\boldsymbol{u}_t|\boldsymbol{q}_t)$.

The chosen action at the current state is evaluated by the cost function $C_t(\boldsymbol{q}_t, \boldsymbol{u}_t) \in \mathbb{R}^1$ and results in a state transition characterized by the probability $P(\boldsymbol{q}_{t+1}|\boldsymbol{q}_t, \boldsymbol{u}_t)$. In Stochastic Optimal Control (SOC), the goal is to find a stochastic policy π^* that minimizes the expected cost

$$\pi^* = \underset{\pi}{\operatorname{argmin}} \left\langle C_T(\boldsymbol{q}_T) + \sum_{t=0}^{T-1} C_t(\boldsymbol{q}_t, \boldsymbol{u}_t) \right\rangle_{q_\pi} \quad , \qquad (1)$$

where the expectation, denoted by the symbols $\langle\cdot\rangle$, is taken with respect to the trajectory distribution

$$q_\pi(\boldsymbol{q}_{0:T}, \boldsymbol{u}_{0:T-1}) = P(\boldsymbol{q}_0) \prod_{t=0}^{T-1} \pi(\boldsymbol{u}_t|\boldsymbol{q}_t) P(\boldsymbol{q}_{t+1}|\boldsymbol{q}_t, \boldsymbol{u}_t) \ ,$$

where $P(\boldsymbol{q}_0)$ is the initial state distribution.

A. Bayesian inference for control

An interesting class of algorithms to SOC problems have been derived by reformulating the original Bellman formulation in (1) as an Bayesian inference problem [14]–[17].

[2]Note that the same principle of regulating the update steps in trajectory optimization can also be applied to planning algorithms in infinite horizon problems such as [11], [12]

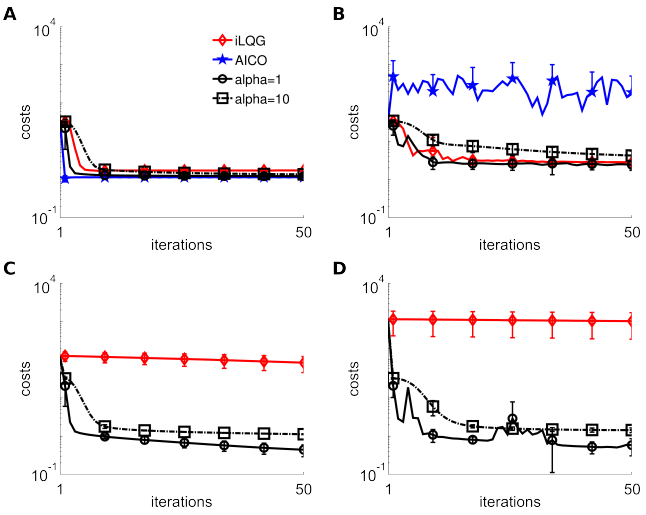

Fig. 2. Comparison of the convergence properties of *iLQG*, AICO and our robust variant, where the rate of convergence is controlled via the parameter α. In the top row (A-B), the model of the forward dynamics was approximated by a *pseudo dynamics* model [13]. In the bottom row, an analytic forward dynamics model of a 5-degree-of-freedom robot arm was used. The panels in the first column denote the costs of the planning algorithms applied to a simple task, where the robot arm has to reach for an end effector target and return to the initial state. In the second column (B,D), the robot has to keep additionally the roll angle constant (at $\pi/2$). Shown are the mean and the standard deviations for 10 initial states '$\boldsymbol{q}_0$ sampled from a Gaussian with zero mean and a standard deviation of 0.05.

Instead of minimizing costs, the idea is to maximize the probability of receiving a reward event ($r_t = 1$) at every time step

$$p(r_t = 1|\boldsymbol{q}_t, \boldsymbol{u}_t) \propto \exp\{-C_t(\boldsymbol{q}_t, \boldsymbol{u}_t)\} \quad . \qquad (2)$$

Note that the idea of turning the cost function in Eq. (1) into a reward signal was also used in operational space control approaches [18], [19].

In the probabilistic framework, we want to compute the posterior over state and control sequences, conditioning on observing a reward at every time step,

$$p_\pi(\boldsymbol{q}_{0:T}, \boldsymbol{u}_{0:T-1}|r_{0:T} = 1) = \exp\{-C_T(\boldsymbol{q}_T)\}$$
$$q_\pi(\boldsymbol{q}_{0:T}, \boldsymbol{u}_{0:T-1}) \prod_{t=1}^{T-1} p(r_t = 1|\boldsymbol{q}_t, \boldsymbol{u}_t) \quad .$$

For efficient implementations of this inference problem, a number of algorithms have been proposed that apply iterative policy updates assuming that all probability distributions can be modeled by an instance of the family of *exponential* distributions [9], [20], [21]. We will restrict our discussion on the Approximate Inference Control (AICO) algorithm with Gaussians [9].

B. Approximate inference control with Gaussians (AICO)

We consider system dynamics of the form $\boldsymbol{q}_{t+1} = f(\boldsymbol{q}_t, \boldsymbol{u}_t) + \epsilon$ with ϵ denoting zero mean Gaussian noise. In AICO (with Gaussians), the system dynamics are linearized through 1st order Taylor expansions, i.e.,

$P(\boldsymbol{q}_{t+1}|\boldsymbol{q}_t, \boldsymbol{u}_t) = \mathcal{N}(\boldsymbol{q}_{t+1}|A_t\boldsymbol{q}_t + a_t + B_t\boldsymbol{u}_t, Q_t)$, where the state transition matrix A_t, the linear drift term a_t and the control matrix B_t are often computed with derivatives simulated through finite differences methods. The numerical stability of AICO also depends on the accuracy of the linearized model, we will therefore additionally compare to an approximation of the system dynamics, where controls $\boldsymbol{u}_t$ correspond directly to joint accelerations[3]. We will refer to this approximation as *pseudo-dynamic* model.

We propose to add a regularization term to the cost function. Before explaining the regularization term in more detail, we briefly discuss how different objectives are implemented in AICO. In the simplest case, the task-likelihood function in (2) can be split into separate state and a control dependent terms, i.e.,

$$p(r_t = 1|\boldsymbol{q}_t, \boldsymbol{u}_t) = \mathcal{N}[\boldsymbol{q}_t|\boldsymbol{r}_t, R_t]\,\mathcal{N}[\boldsymbol{u}_t|\boldsymbol{h}_t, H_t] \quad , \quad (3)$$

where, for analytical reasons, the Gaussians are given in *canonical* form, i.e., $\mathcal{N}[\boldsymbol{u}_t|\boldsymbol{h}_t, H_t] \propto \exp(-1/2\,\boldsymbol{u}_t^T H_t \boldsymbol{u}_t + \boldsymbol{u}_t^T \boldsymbol{h}_t)$. Note that the vector $\boldsymbol{r}_t$ in (3) denotes the linear term for the Gaussian distribution and must not be confused with the auxiliary variable $r_t = 1$ in (2) denoting a reward event. By inserting (3) in (2) we obtain the quadratic costs,

$$C_t(\boldsymbol{q}_t, \boldsymbol{u}_t) = \boldsymbol{q}_t^T R_t \boldsymbol{q}_t - 2\boldsymbol{r}_t^T \boldsymbol{q}_t + \boldsymbol{u}_t^T H \boldsymbol{u}_t - 2\boldsymbol{h}_t^T \boldsymbol{u}_t \quad . \quad (4)$$

The state dependent costs, encoded by $\mathcal{N}[\boldsymbol{q}_t|\boldsymbol{r}_t, R_t]$, can be defined in configuration space[4], in task space[5], or even in combinations of both spaces [16].

On the algorithmic level, AICO combines forward messages and backward messages to compute the belief over trajectories. We represent these Gaussian forward message by $N[\boldsymbol{q}_t|\boldsymbol{s}_t, S_t]$, the backward message by $N[\boldsymbol{q}_t|\boldsymbol{v}_t, V_t]$, and the belief by $N[\boldsymbol{q}_t|\boldsymbol{b}_t, B_t]$. The recursive update equations are given in [9] and in [10] where an implementation which additionally implements control constraints (otherwise $\boldsymbol{h}_t = 0$) is given.

We can also compute the most likely action given the task constraints. By doing so, in the case of AICO with Gaussians, we obtain a time varying linear feedback controller

$$\boldsymbol{u}_t^{[n]} = \boldsymbol{o}_t + O_t \boldsymbol{q}_t \quad , \quad (5)$$

where $\boldsymbol{o}_t$ is an open loop gain and O_t denotes the feedback gain matrix (n denotes the iteration).

[3]For a single joint with $\boldsymbol{q} = [q, \dot{q}]^T$, the matrix $A = \left(\begin{smallmatrix} 1 & \tau \\ 0 & 1 \end{smallmatrix}\right)$, $a = \left(\begin{smallmatrix} 0 \\ 0 \end{smallmatrix}\right)$, and $B = \left(\begin{smallmatrix} \tau^2 \\ \tau \end{smallmatrix}\right)$, where τ denotes the time step.

[4]Reaching a goal state $\boldsymbol{g}^* \in \mathbb{Q}$ in configuration space can be encoded by $\boldsymbol{r}_t = R_t \boldsymbol{g}^*$ where the precision matrix R_t scales the importance of different dimensions.

[5]Let $\boldsymbol{x}^* \in \mathbb{X}$ denote a desired end effector position and let $\boldsymbol{x} = f(\boldsymbol{q})$ be the forward kinematics mapping and $J(\boldsymbol{q}_t) = \partial f / \partial \boldsymbol{q} | \boldsymbol{q} = \boldsymbol{q}_t$ its Jacobian. We can now obtain a Gaussian task likelihood by approximating the forward kinematics by its linearization through the Jacobian, i.e., $\boldsymbol{x} \approx f(\boldsymbol{q}_0) + J(\boldsymbol{q} - \boldsymbol{q}_0)$. The parameters of the Gaussian are then given by $\boldsymbol{r}_t = J^T C \big(f(\boldsymbol{q}_0) - \boldsymbol{x}^*\big)$ and $R_t = J^T C J$, where the diagonal elements of the matrix C specify the desired precision in task space.

Algorithm 1: Approximate Inference Control with Regularized Update Steps

Input: initial state $\boldsymbol{q}_0$, parameter $\alpha^{[0]}$, threshold θ
Output: feedback control law $\boldsymbol{o}_{0:T-1}$ and $O_{0:T-1}$
initialize $\boldsymbol{q}_{1:T}^{[0]} = \boldsymbol{q}_0$, $S_0 = 1e10 \cdot \mathbf{I}$, $\boldsymbol{s}_0 = S_0 \boldsymbol{q}_0$, $n = 1$
while *not converged* **do**
  $\boldsymbol{q}_{0:T}^{[n-1]} = \boldsymbol{q}_{0:T}^{[n]}$
  for $t \leftarrow 1$ **to** T **do**
    linearize model: $A_t, \boldsymbol{a}_t, B_t$
    compute: $H_t, \boldsymbol{h}_t, R_t, \boldsymbol{r}_t$
    update: $\boldsymbol{s}_t$, S_t, $\boldsymbol{v}_t$, V_t, $\boldsymbol{b}_t$, and B_t
    if $\|\boldsymbol{b}_t - \boldsymbol{q}_t^{[n]}\| > \theta$ **then**
      repeat this time step
      $t \leftarrow t - 1$
    $\boldsymbol{q}_t^{[n]} = B_t^{-1} \boldsymbol{b}_t$
  for $t \leftarrow T - 1$ **to** 0 **do**
    ..same updates as above...
  for $t \leftarrow 0$ **to** $T - 1$ **do**
    compute feedback controller: $\boldsymbol{o}_t, O_t$
    $\boldsymbol{u}_t^{[n]} = \boldsymbol{o}_t + O_t \boldsymbol{q}_t$
    $\boldsymbol{q}_{t+1}^{[n]} = A_t \boldsymbol{q}_t^{[n]} + \boldsymbol{a}_t + B_t \boldsymbol{u}_t^{[n]}$
  $n = n + 1$
  $\alpha^{[n]} = \alpha^{[n-1]} \gamma$
return $\boldsymbol{o}_{0:T-1}$ and $O_{0:T-1}$

C. Evaluation of the convergence properties of AICO

To investigate the convergence properties of AICO, we use a simulated light-weight robot arm [22] with five joints. The robot has to reach a desired end effector position in Cartesian space and subsequently has to return to its initial pose. To increase the complexity, we define a second task, where the robot should additionally keep the roll angle of the end effector constant. For this task, we used the cost function

$$C_t(\boldsymbol{q}_t, \boldsymbol{u}_t) = \begin{cases} 10^4 (\boldsymbol{x}^i - \boldsymbol{x}_t)^T (\boldsymbol{x}^i - \boldsymbol{x}_t) & \text{if } t = T^i \\ 10^{-2} \boldsymbol{u}^T \boldsymbol{u} & \text{else} \end{cases} \quad , \quad (6)$$

where $\boldsymbol{x}^i$ denotes the desired robot postures in task space at times $T^1 = 500$ and $T^2 = 10^3$ (the planning horizon is 2 seconds with a time step of 2ms) with $\boldsymbol{x}^1 = [1, -0.4, 0, 0, \pi/2, 0]^T$ and $\boldsymbol{x}^2 = [1, 0, 0, 0, \pi/2, 0]^T$. Note that we do not assume any initial solution to initialize the planner, solely the initial posture of the robot in configuration space is used as initial 'trajectory'. An example movement is shown in Figure 1.

Using the *pseudo-dynamics* approximation of the system dynamics, the convergence rate of the costs per iteration of both tasks are shown in Figure 2A,B. For the simple task in Figure 2A the inferred cost values converge fast for all algorithms, with the standard AICO algorithm showing

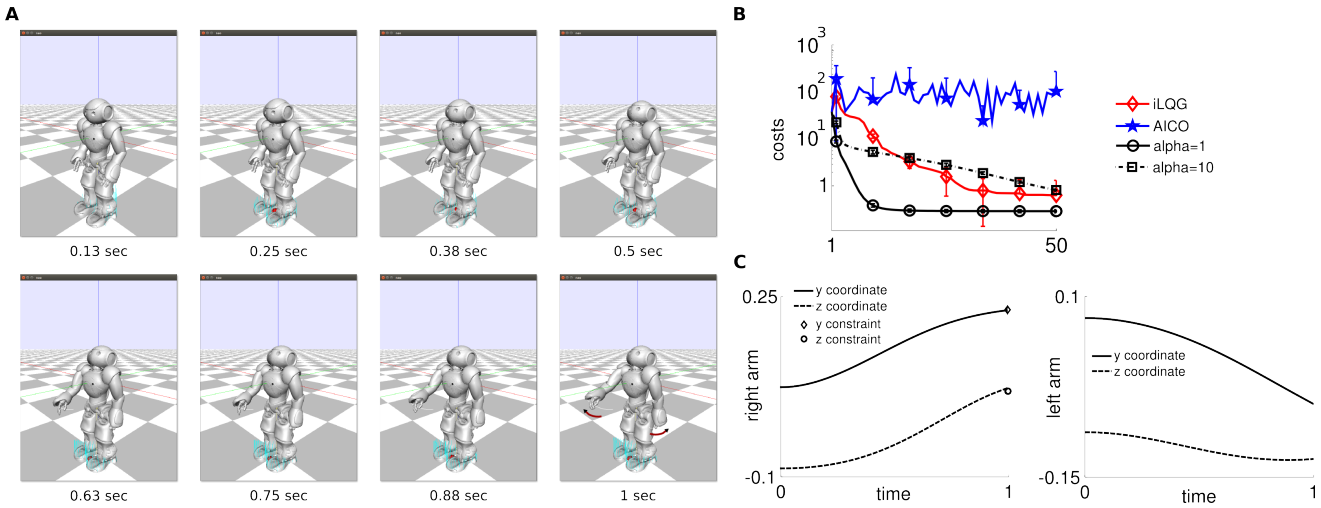

Fig. 3. Reaching task with the humanoid robot *Nao*. The robot has to reach a desired end effector position with the right arm while maintaining balance. Eight snapshots of the inferred movement are shown in (A). In (B), the convergence of the costs of the optimization procedure is shown, where we compare *iLQG*, the standard implementation of AICO and the regularized variant. The mean and the standard deviations for 10 initial states '$\boldsymbol{q}_0$ are sampled from a Gaussian with zero mean and a standard deviation of 0.05. The movement objectives for the right arm are shown in the left panel in (C). To counter balance, the robot moves its left hand and the head.

the best performance. However, the fast convergence also comes with the costs of a reduced robustness of the policy update as can be seen from the results in the second scenario illustrated in Figure 2B, where AICO is unstable and cannot infer solutions with low costs. When we used the analytic forward dynamic model (where the linearizations are computed through finite differences) instead of the pseudo dynamics model, computing the messages in AICO became numerically unstable and no solutions could be inferred. Therefore, the panels in Figure 2C,D do not include results of AICO. We also evaluated the *iLQG* method [7] that implements an adaptive regularization schedule and line search to prevent divergence [8]. While the *iLQG* algorithm performed well for the pseudo dynamics model, the learning rate was automatically decreased to almost zero for the analytical dynamics model. Our regularization method for AICO, that we will present in the next section, considerably outperformed both competing methods.

D. Regulating the policy updates in AICO

To regularize the policy update steps in (1), we add an additional cost term to the task-likelihood function, i.e.,

$$p(r_t = 1|\boldsymbol{q}_t^{[n]}, \boldsymbol{u}_t^{[n]}) \propto \exp\{-C_t(\boldsymbol{q}_t^{[n]}, \boldsymbol{u}_t^{[n]}) - \alpha^{[n]}(\boldsymbol{q}_t^{[n]} - \boldsymbol{q}_t^{[n-1]})^T(\boldsymbol{q}_t^{[n]} - \boldsymbol{q}_t^{[n-1]})\} \quad ,$$

which punishes the distance of the state trajectories of two successive iterations of the algorithm (n denotes the iteration). The parameter α controls the size of the update step. For large α, the trajectory update will be conservative as the algorithm will stay close to the previous trajectory that has been used for linearization. For small α values, the new trajectory will directly jump to the LQG solution given the linearized dynamics and the approximated costs. Hence, α is inverse proportional to the step size. The value of α is updated after each iteration according to $\alpha^{[n]} = \alpha^{[n-1]}\gamma$. For $\alpha^{[0]} \geq 1$ and $\gamma > 1$, convergence is guaranteed as the regularization term will dominate with an increasing number of iterations.

The algorithms is listed in Algorithm 1. An interesting feature of this algorithm is that no learning rate is needed as α is used directly to implement a step size. In the original formulation of AICO the learning rate is either applied to the state update (in Line 13 in Algorithm 1) [9] or to the feedback controller (in Line 18 in Algorithm 1) [10]. However, neither implementation can guarantee convergence in nonlinear systems or in tasks with costs inducing a nonlinear mapping from $\mathbb{Q}$ to $\mathbb{X}$.

We evaluate the resulting algorithm on the same robot arm reaching tasks. For both tasks, the Cartesian planning task in Figure 2A,C and the extension with the additional roll angle objective in Figure 2B,D, we evaluated AICO with the regularization parameter $\alpha \in \{1, 10\}$ (we did not increase α and $\gamma = 1$). For both models of the system dynamics, the *pseudo-dynamics* approximation (shown in Figure 2A,B) and the analytic model (illustrated in Figure 2C,D), AICO benefits from the regularization term and the costs decay exponentially fast. Interestingly, without "good" initial solutions, the differential dynamic programming method *iLQG* [8] that implements a sophisticated regularization scheme cannot generate movement policies with low costs when using the analytic model. This is shown in Figure 2C,D.

III. RESULTS

We evaluated the proposed planning method in simulation with the humanoid robot Nao. The Nao robot has 25 degrees-of-freedom. In first experiments, we investigated the performance of the planner with a *pseudo-dynamics* model

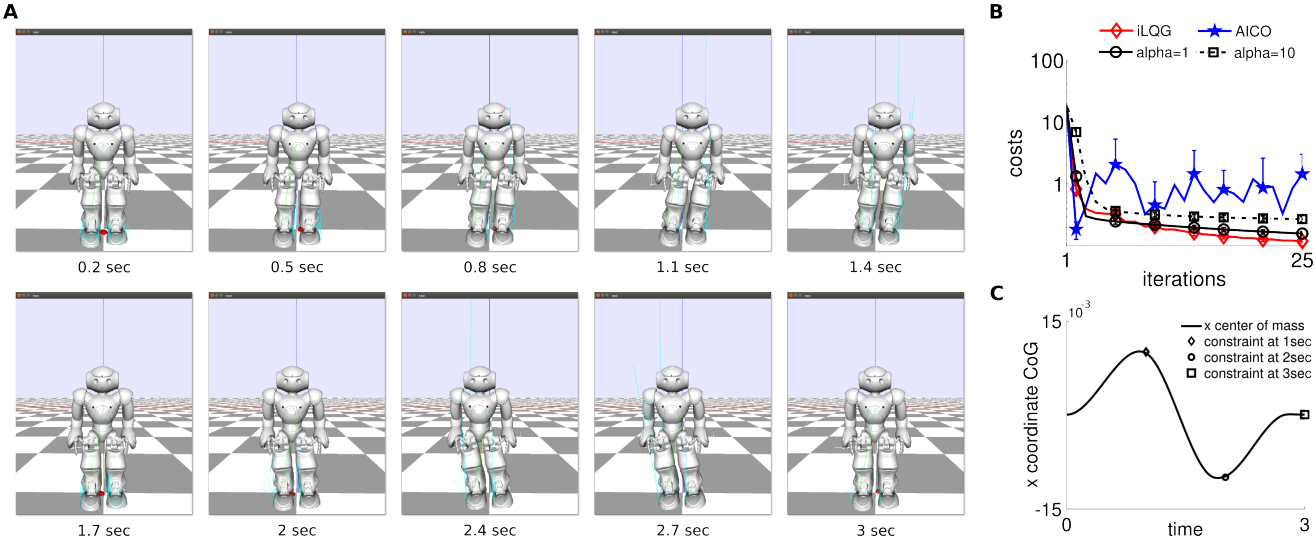

Fig. 4. Balancing task in the humanoid robot *Nao*. The robot should *swing* its hips, which is encoded by adding an offset scalar to the x-coordinate of the center of gravity vector. In (A) 10 snapshots of the resulting movement for an increasing planning horizon are shown for $\alpha = 1$. The convergence properties of *iLQG*, the standard AICO and its regularized variants are shown in (B). The mean and the standard deviations for 10 initial states '$\boldsymbol{q}_0$ are sampled from a Gaussian with zero mean and a standard deviation of 0.05. In (C) the x-coordinate of the center of gravity of the Nao is illustrated. The large dots denote the objectives.

of the robot.

The humanoid had to reach for an end effector target using the right arm while maintaining balance. In a second experiment, Nao had to shift the x-coordinate of the center of gravity while maintaining balance.

A. Arm reaching with a humanoid robot

The humanoid has to reach for the end effector target $\boldsymbol{x}^* = [0, 0.2, 0.06]^T$, where only the y- and the z- Cartesian coordinates are relevant. Additionally, the robot has to maintain balance, which is implemented as deviation of the center of gravity vectors from its initial values $\boldsymbol{x}_{\text{CoG}}(t = 0)$, i.e., we specify the desired center of gravity as $\boldsymbol{x}^*_{\text{CoG}} = \boldsymbol{x}_{\text{CoG}}(t = 0)$. The same cost function as in the experiments for the light weight robot arm in (6) is used. For this task, however, only a single via-point is defined that is used for the desired end effector target and the center of gravity, i.e., $\boldsymbol{x}^1 = [\boldsymbol{x}^{*T}, \boldsymbol{x}^*_{\text{CoG}}{}^T]^T$.

Only by specifying two scalars in $\boldsymbol{x}^*$ (the scaling parameters in (6) are constants that take the values 10^4 or 10^{-2}), the planning algorithm infers 50-dimensional state trajectories (the state $\boldsymbol{q}_t$ at time t encodes the joint angles and the joint velocities, ignoring the base frame). This is shown in Figure 3A for the proposed planning algorithm with the regularization parameter $\alpha = 1$. As in the robot arm experiments, the Approximate Inference Control (AICO) algorithm benefits from the regularization. As can be seen in Figure 3B, AICO cannot infer movement solutions with low costs without regularization.

Interestingly, to maintain balance, the humanoid utilizes its head and its left arm for which no objectives were explicitly specified. This effect is a feature of model-based planning methods that consider the coupled dynamics and is best illustrated in Figure 3C, where the end effector trajectories of both arms and the desired target values are shown.

B. Balancing with a humanoid

In this task the humanoid has to balance on one foot by moving its center of gravity. In this experiment, we specify three desired via-points for the center of gravity, i.e., $\boldsymbol{x}^i = \boldsymbol{x}^i_{\text{CoG}}$ with $i = 1, ..., 3$. The last via-point is set to the initial center of gravity $\boldsymbol{x}_{\text{CoG}}(t = 0)$. The first via-point has an offset of 0.1m in the x-coordinate of $\boldsymbol{x}_{\text{CoG}}(t = 0)$ to force the robot to move its center of gravity to the right. The second via point has the same negative offset in the x-direction to exhibit a movement to the left. The planning horizon was three seconds ($T^1 = 100$,$T^2 = 200$ and $T^3 = 300$ with $\tau = 10$ms) and the distance matrix C in (6) was scaled with the importance weights $[10^6, 10, 10]^T$ for the x,y, and z coordinate of $\boldsymbol{x}^i_{\text{CoG}}$.

For $\alpha = 1$, the resulting movement is illustrated in Figure 4A. Illustrated are 10 snapshots. Nao first moves its hip to the right (with respect to the robot frame) and thereafter to the left. This movement is the result of an inference problem encoded in mainly two scalars, i.e., the offsets.

The standard implementation of AICO was not able to infer successful balancing solutions, which is illustrated in Figure 4B. In contrast, the regularized variant using $\alpha \in \{1, 10\}$ converged after 25 iterations of the trajectory optimization procedure. For $\alpha = 1$, the x-coordinate of the center of gravity and the implemented objectives are shown in 4C.

C. Computational time

The computational time of the proposed planning algorithm is the same as for the standard implementation of

AICO. If the algorithm is implemented in C-code it achieves real time performance in humanoid planning problems [9]. However, for our experiments we used a Matlab implementation on a standard computer (2.4GHz, 8GB RAM), where, e.g., the computation of the balancing movements in Figure 4 took less then 50 seconds (which includes all 25 iterations of the optimization process). The movement duration of the executed trajectory was three seconds.

IV. CONCLUSIONS

Stochastic Optimal Control (SOC) methods are powerful planning methods to infer high-dimensional state and control sequences [7]–[9], [20]. For real time applications in humanoids, efficient model predictive control variants have been proposed [8]. However, the quality of the generated solutions heavily depends on the initial movement policy and on the accuracy of the approximations of the system dynamics. Most methods use regularization to prevent numerical instabilities, but typically greedily exploit the approximated system dynamics model. The resulting trajectory update might be far from the previous trajectory used for linearization.

As the linearizations are only locally valid, we explicitly avoid large jumps in the trajectories by punishing large deviations from the previous trajectory. We demonstrated in this paper that SOC methods can greatly benefit from such a regularization term. We used such regularization term for the Approximate Inference Control (AICO) algorithm [9]. Due to the regularization term, which implicitly specifies the step size of the trajectory update, no learning rate as in the standard formulation of AICO is needed. Our experiment shows that the used regularization term considerably outperforms existing SOC methods that are based on linearization, in particular if highly non-linear system dynamics are used.

An interesting open question is if the proposed regularization parameter facilitates a combination of SOC and model learning approaches. Typically, inaccurate model predictions have catastrophic effects on the numerical stability of SOC methods. In particular, if the model predictions are poor, the SOC method should not further explore but collect more data around the current trajectory. Such idea could be implemented by modeling the regularization parameter as a function of the model uncertainty.

ACKNOWLEDGMENT

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreements No. 270327 (CompLACS) and No. 600716 (CoDyCo). The authors would like to acknowledge Guilherme Maeda, Tucker Hermans and Serena Ivaldi for their assistance during the preparation of this manuscript.

REFERENCES

[1] Paolo Baerlocher and Ronan Boulic. Task-priority formulations for the kinematic control of highly redundant articulated structures. In *IROS*, pages 13–17, 1998.

[2] Su Il Choi and Byung Kook Kim. Obstacle avoidance control for redundant manipulators using collidability measure. *Robotica*, pages 143–151, 2000.

[3] Tomomichi Sugihara and Yoshihiko Nakamura. Whole-body cooperative balancing of humanoid robot using cog jacobian. In *IROS*, pages 2575–2580, 2002.

[4] Michael Gienger, Herbert Janssen, and Christian Goerick. Task-oriented whole body motion for humanoid robots. In *IEEE-RAS*, pages 238–244, 2005.

[5] Koichi Nishiwaki, Mamoru Kuga, Satoshi Kagami, Masayuki Inaba, and Hirochika Inoue. Whole-body cooperative balanced motion generation for reaching. *IJHR*, pages 437–457, 2005.

[6] Robert F Stengel. *Stochastic optimal control: theory and application*. John Wiley & Sons, Inc., 1986.

[7] Emanuel Todorov and Weiwei Li. A generalized iterative lqg method for locally-optimal feedback control of constrained nonlinear stochastic systems. In *ACC*, pages 300–306, 2005.

[8] Yuval Tassa, Tom Erez, and Emanuel Todorov. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *IROS*, pages 4906–4913, 2012.

[9] Marc Toussaint. Robot trajectory optimization using approximate inference. In *ICML*, pages 1049–1056, 2009.

[10] Elmar Rueckert and Gerhard Neumann. Stochastic optimal control methods for investigating the power of morphological computation. *Artificial Life*, pages 115–131, 2013.

[11] Marc Toussaint and Amos Storkey. Probabilistic inference for solving discrete and continuous state markov decision processes. In *ICML*, pages 945–952, 2006.

[12] Matthew D Hoffman, Nando D Freitas, Arnaud Doucet, and Jan R Peters. An expectation maximization algorithm for continuous markov decision processes with arbitrary reward. In *AISTATS*, pages 232–239, 2009.

[13] Marc Toussaint, Nils Plath, Tobias Lang, and Nikolay Jetchev. Integrated motor control, planning, grasping and high-level reasoning in a blocks world using probabilistic inference. In *ICRA*, pages 385–391, 2010.

[14] Marc Toussaint and Christian Goerick. Probabilistic inference for structured planning in robotics. In *Int. Conf. on Intelligent Robots and Systems (IROS 2007)*, pages 3068–3073, 2007.

[15] Thomas Furmston and David Barber. Variational methods for reinforcement learning. In *AISTATS*, pages 241–248, 2010.

[16] Marc Toussaint and Christian Goerick. A bayesian view on motor control and planning. In *From Motor Learning to Interaction Learning in Robots*, pages 227–252. Springer, 2010.

[17] Hilbert J Kappen, Vicenç Gómez, and Manfred Opper. Optimal control as a graphical model inference problem. *JMLR*, pages 159–182, 2012.

[18] J. Peters and S. Schaal. Learning operational space control. In *Proceedings of Robotics: Science and Systems*, Philadelphia, USA, August 2006.

[19] Jan Peters and Stefan Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *ICML*, pages 745–750, 2007.

[20] Konrad Rawlik, Marc Toussaint, and Sethu Vijayakumar. An approximate inference approach to temporal optimization in optimal control. In *NIPS*, pages 2011–2019, 2010.

[21] Jur Van Den Berg, Sachin Patil, and Ron Alterovitz. Motion planning under uncertainty using iterative local optimization in belief space. *IJRR*, pages 1263–1278, 2012.

[22] Thomas Lens, Jürgen Kunz, Oskar von Stryk, Christian Trommer, and Andreas Karguth. Biorob-arm: A quickly deployable and intrinsically safe, light-weight robot arm for service robotics applications. In *ISR/ROBOTIK*, pages 1–6, 2010.

Chapter 5

Recurrent Spiking Networks Solve Planning Tasks (TUD)

Recurrent Spiking Networks Solve Planning Tasks

Elmar Rueckert[1,*], David Kappel[2], Daniel Tanneberg[1], Dejan Pecevski[2], and Jan Peters[1,3]

[1]Intelligent Autonomous Systems Lab, Technische Universität Darmstadt, 64289, Germany
[2]Institute for Theoretical Computer Science, Technische Universität Graz, 8020, Austria
[3]Robot Learning Group, Max-Planck Institute for Intelligent Systems, Tuebingen, 72076, Germany
[*]rueckert@ias.tu-darmstadt.de

ABSTRACT

A recurrent spiking neural network is proposed that implements planning as probabilistic inference for finite and infinite horizon tasks. The architecture splits this problem into two parts: The stochastic transient firing of the network embodies the dynamics of the planning task. With appropriate injected input this dynamics is shaped to generate high-reward state trajectories. A general class of reward-modulated plasticity rules for these afferent synapses is presented. The updates optimize the likelihood of getting a reward through a variant of an Expectation Maximization algorithm and learning is guaranteed to convergence to a local maximum. We find that the network dynamics are qualitatively similar to transient firing patterns during planning and foraging in the hippocampus of awake behaving rats. The model extends classical attractor models and provides a testable prediction on identifying modulating contextual information. In a real robot arm reaching and obstacle avoidance task the ability to represent multiple task solutions is investigated. The neural planning method with its local update rules provides the basis for future neuromorphic hardware implementations with promising potentials like large data processing abilities and early initiation of strategies to avoid dangerous situations in robot co-worker scenarios.

Introduction

Probabilistic inference has emerged as a promising framework for solving Markov decision problems (*probabilistic planning*).[1,2] In parallel to this development, the probabilistic inference perspective has also been successfully used in cognitive science and neuroscience for modeling how biological organisms solve planning problems.[3–6] However, it was not clear how probabilistic planning can be implemented in neural substrates with biologically realistic learning rules. Here, we show that probabilistic planning can be implemented and learned by recurrent networks of spiking neurons such that the generated spike sequences realize mental plans.

Recently, it was shown that recurrent spiking networks can implement *Bayesian filtering* and are able to learn a generative model of temporal sequences from presented examples drawn from a target distribution (i.e., the distribution that has generated the samples).[?,7,8] Training was realized through synaptic plasticity rules without supervision. After learning, when running freely, the neural networks effectively perform forward sampling in the dynamic Bayesian network representing the target distribution. For solving a *planning* problem, however, it is necessary to sample from the posterior distribution conditioned on receiving a reward. In other words, forward sampling needs to integrate future rewards propagating backward in time in the dynamic Bayesian network.

We consider a recurrent network of stochastic spiking neurons, which without any input, samples sequences through forward sampling from the corresponding dynamic Bayesian network. We show that by injecting appropriate input in the network from a layer of task-related *context neurons*, it can generate samples from the posterior distribution conditioned on getting a reward. Optimal reward-modulated Hebbian learning rules are derived that implement planning as probabilistic inference in

the spiking network through iterative local updates. The recurrent dynamics of the neural network can be reused for solving different planning problems through activating different sets of context neurons which encode different goals and constraints (e.g., reaching for different goals or avoiding obstacles).

The presented theory is compatible with recent results on transient firing observed in behaving animals in phases of rest.[9–12] In particular, the model reproduces characteristic spatiotemporal spiking activity that preplays movement paths to remembered home locations[9] and provides an alternative to attractor networks,[13] which are typically used to explain movement planning in maze navigation tasks.[14–17] In attractor networks only the end point can be adapted. For non-straight line paths, for example to escape a labyrinth, complex attractor sequence models were proposed.[18] The presented neural model allows to learn such non-straight line paths from single scalar rewards that encode in addition to desired goals and obstacles knowledge about rewarding routes. The ability to modulate the shape of a movement path is a testable prediction for future neuroscience studies.

Another contribution of the presented theory is that it provides the basis for neuromorphic hardware implementations of control and planning strategies in mobile robots. Similar architectures building also on winner-take-all circuits and spike-timing dependent plasticity (STDP) rules were already proposed as computational models for such brain-like chips.[?,?,?] For planning, neuromorphic hardware implementations promise to process large input streams from visual and tactile sensors through parallel computing,[?] go round strategies may be initiate in real-time to avoid dangerous situations in robot co-worker scenarios and event based neural network implementations are energy efficient alternatives to classical von Neumann architectures.[?] In first experiments, we demonstrate that spiking neural networks can be trained to simultaneously represent multiple obstacle avoiding strategies in a real robot arm reaching task.

Methods

We first formulate finite horizon probabilistic planning tasks with terminal rewards only. Later we will generalize to infinite horizons where rewards can be received at any point in time. Let $\boldsymbol{x}_t \in \mathscr{R}^d$ denote the d dimensional continuous state of a behaving agent at time t. The goal of the agent is to find the sequence of states $\underline{\boldsymbol{x}} = (\boldsymbol{x}_1 \ldots \boldsymbol{x}_T)$ that maximizes the total received reward (i.e., the return) $\hat{r}$ at the end of the trial.

Such planning tasks can be modeled as inference problems[1,2] where the joint distribution over state sequences and returns is given by $p(\underline{\boldsymbol{x}}, r) = p(\boldsymbol{x}_0)\, p(r \mid \underline{\boldsymbol{x}}) \prod_{t=1}^{T} \mathscr{T}(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1})$. The distribution $p(\boldsymbol{x}_0)$ encodes an initial prior over states, $\mathscr{T}$ corresponds to the state transition model, and the distribution $p(r \mid \underline{\boldsymbol{x}})$ determines the probability of receiving the return r given the trajectory of states $\underline{\boldsymbol{x}}$. As in related work[1,2,19] we assume that $p(r \mid \underline{\boldsymbol{x}})$ can be factorized as $p(r \mid \underline{\boldsymbol{x}}) = 1/Z_r \prod_{t=1}^{T} \psi_t(r|\boldsymbol{x}_t)$. In this formulation, r denotes a binary random variable, where without loss of generality, the probability of observing such a binary return event is a modulo rescaling of the original reward maximization problem.[1]

An agent can use such an internal model of the environment to plan a sequence of movements by solving the inference

problem

$$p\left(\underline{\boldsymbol{x}} \mid r=1\right)=\frac{1}{\mathscr{Z}} p\left(r \mid \underline{\boldsymbol{x}}\right) p\left(\boldsymbol{x}_0\right) \prod_{t=1}^{T} \mathscr{T}\left(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}\right) \quad , \tag{1}$$

where $\mathscr{Z}=\sum_{\underline{\boldsymbol{x}}} p\left(\underline{\boldsymbol{x}}, r\right)$ is a term that guarantees that equation (1) is normalized. In our formulation of the planning problem, the actions in the probabilistic model are integrated out. It is assumed that the actions can be subsequently inferred from the posterior over state sequences.

The unconstrained process for planning models a freely moving agent by

$$p\left(\underline{\boldsymbol{x}}\right)=p\left(\boldsymbol{x}_0\right) \prod_{t=1}^{T} \mathscr{T}\left(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}\right) \quad . \tag{2}$$

Sampling from this probability distribution can be implemented by a recurrent network of spiking neurons (e.g., using ideas from[?,7,8]). However, it is not straightforward for a recurrent network to solve the inference problem in equation (1), which requires to integrate future returns backward in time. Only local temporal information is available when sampling from the network. Such temporal models are different to model-based Markov decision process methods encoding global value or Q functions.[20]

We propose here a solution to this problem that relies on replacing $p\left(\underline{\boldsymbol{x}}\right)$ with a model distribution $q\left(\underline{\boldsymbol{\nu}}; \boldsymbol{\theta}\right)$, where sampling from $q\left(\underline{\boldsymbol{\nu}}; \boldsymbol{\theta}\right)$ is implemented with an extended neural network architecture and $\underline{\boldsymbol{\nu}}$ is the neural approximation of $\underline{\boldsymbol{x}}$. The parameters $\boldsymbol{\theta}$ are learned such that the Kullback-Leibler divergence between the true posterior for planning in equation (1) and a model distribution $q\left(\underline{\boldsymbol{\nu}}; \boldsymbol{\theta}\right)$ converges to zero.

Planning with recurrent neural networks

We propose a recurrent spiking neural network to implement planning. Our network consists of two populations of neurons, which we denote by Y and V (see Fig. 1A). V is a layer of K *state neurons* that control the state (e.g., the agent's spatial location) of a freely moving agent. These neurons receive lateral connections from neighboring state neurons and from all N neurons in a population of *context neurons* Y with weights w_{ki} and θ_{kj}. The context neurons produce spatiotemporal spike patterns that represent high-level goals and context information (e.g., the target state that should be reached after T time steps). We show that probabilistic planning problems defined in equation (1) can be implemented in the network by training the synapses θ_{kj}.

We denote the activity of the state neurons at time t by $\boldsymbol{\nu}_t=\left(\nu_{t,1}, \ldots, \nu_{t,K}\right)$, where $\nu_{t,k}=1$ if neuron k spiked at time t and $\nu_{t,k}=0$ else. Discrete random variables $\boldsymbol{x}_t$ can be encoded as a multinomial distribution, where one neuron maps to one state instance. For continuous variables a simple encoding scheme is used, i.e., $\boldsymbol{x}_t=1/\left|\boldsymbol{\nu}_t\right| \sum_{k=1}^{K}\left(\nu_{t,k} \boldsymbol{p}_k\right)$, where $\left|\boldsymbol{\nu}_t\right|=\sum_{k=1}^{K} \nu_{t,k}$ and $\boldsymbol{p}_k$ is the preferred position of state neuron k.

Analogously, we define the spiking activity of the context neurons at time t by a binary vector $\boldsymbol{y}_t=\left(y_{t,1}, \ldots, y_{t,N}\right)$. Using

these definitions of $\boldsymbol{\nu}_t$ and $\boldsymbol{y}_t$, we define the membrane potential $u_{t,k}$ and firing probability $\rho_{t,k}$ of state neuron k at time t by

$$u_{t,k} = \sum_{i=1}^{K} w_{ki}\,\nu_{t-1,i} + \sum_{j=1}^{N} \theta_{kj}\,y_{t-1,j} \quad \text{and} \quad \rho_{t,k} = p\left(\nu_{t,k}=1\right) = f(u_{t,k})\,. \tag{3}$$

The last term $f(u_{t,k})$ denotes the activation function, where we only require that it is differentiable. The probability that the network generates a spike sequences $\underline{\boldsymbol{\nu}} = (\boldsymbol{\nu}_1, \ldots, \boldsymbol{\nu}_T)$ of length T starting from a given initial state $\boldsymbol{\nu}_0$ is thus

$$q\left(\underline{\boldsymbol{\nu}};\boldsymbol{\theta}\right) = p(\boldsymbol{\nu}_0)\prod_{t=1}^{T}\prod_{k=1}^{K} {\rho_{t,k}}^{\nu_{t,k}}\left(1-\rho_{t,k}\right)^{1-\nu_{t,k}}\,. \tag{4}$$

We assume that the transition model (encoded in the synaptic weights w_{ki}) is known or was acquired in a pre-learning phase, e.g., using contrastive divergence learning.[21] Using this assumption, we define the goal of probabilistic planning as minimizing the Kullback-Leibler divergence between the true posterior for planning in equation (1) and the model distribution

$$D_{KL}(p(\underline{\boldsymbol{\nu}}|r=1)\,||\,q\left(\underline{\boldsymbol{\nu}};\boldsymbol{\theta}\right)) = \sum_{\underline{\boldsymbol{\nu}}} p(\underline{\boldsymbol{\nu}}|r=1)\log\frac{p(\underline{\boldsymbol{\nu}}|r=1)}{q\left(\underline{\boldsymbol{\nu}};\boldsymbol{\theta}\right)} = -H_{p(\underline{\boldsymbol{\nu}}|r=1)} - \left\langle \log q\left(\underline{\boldsymbol{\nu}};\boldsymbol{\theta}\right)\right\rangle_{p(\underline{\boldsymbol{\nu}}|r=1)}\,, \tag{5}$$

where $H_{p(\underline{\boldsymbol{\nu}}|r=1)}$ denotes the entropy of the true data distribution. Thus, solving the inference problem in equation (1) is equal to minimizing the Kullback-Leibler divergence in equation (5).

Typically, $p(\underline{\boldsymbol{\nu}}|r=1)$ is unknown and we cannot draw samples from it. However, we can draw samples from the model distribution $q\left(\underline{\boldsymbol{\nu}};\boldsymbol{\theta}\right)$ and update the parameters such that the probability of receiving a reward event is maximized

$$\Delta\theta_{kj} = \eta\left\langle r\frac{\partial}{\partial\theta_{kj}}\log q\left(\underline{\boldsymbol{\nu}};\boldsymbol{\theta}\right)\right\rangle_{p(r\,|\,\underline{\boldsymbol{\nu}})\,q(\underline{\boldsymbol{\nu}};\boldsymbol{\theta})}\,, \tag{6}$$

where η denotes a small learning rate. Note that this general update rule is the result of a standard maximum likelihood formulation where we exploited that $p(\underline{\boldsymbol{\nu}}|r=1) \propto p\left(r\,|\,\underline{\boldsymbol{\nu}}\right)p\left(\underline{\boldsymbol{\nu}}\right)$ using equation (1) and equation (2). The update is an instance of the Expectation-Maximization (EM) algorithm,[22] where evaluating the expectation with respect to $p\left(r\,|\,\underline{\boldsymbol{\nu}}\right)q\left(\underline{\boldsymbol{\nu}};\boldsymbol{\theta}\right)$ corresponds to the E-step and the parameter update realizes the M-step. The update is also related to policy gradient methods[?,?,?] with the difference that we interpret the parameters $\boldsymbol{\theta}$ as having the role of linear controls.[23]

To derive the update rule for the proposed neural network architecture, the network dynamics in equation (4) is used in equation (6). For a detailed derivation we refer to the supplement. The spiking network update rule reads

$$\Delta\theta_{kj} = \eta\left\langle r\sum_{t=1}^{T}\left(\nu_{t,k}-\rho_{t,k}\right)\frac{\partial}{\partial\theta_{kj}}\operatorname{logit}(\rho_{t,k})\right\rangle_{p(r\,|\,\underline{\boldsymbol{\nu}})\,q(\underline{\boldsymbol{\nu}};\boldsymbol{\theta})}\,, \tag{7}$$

where $\operatorname{logit}(\rho_{t,k}) = \log\left(\rho_{t,k}/(1-\rho_{t,k})\right)$ are the log-odds of neuron k firing at time t. Equation (7) is the general learning rule for arbitrary differentiable activation functions $f(u_{t,k}) = \rho_{t,k}$. It adapts the weights θ_{kj} to maximize the return. For many relevant activation functions (e.g., exponential or sigmoid functions), equation (7) turns into a simple reward-modulated Hebbian-type

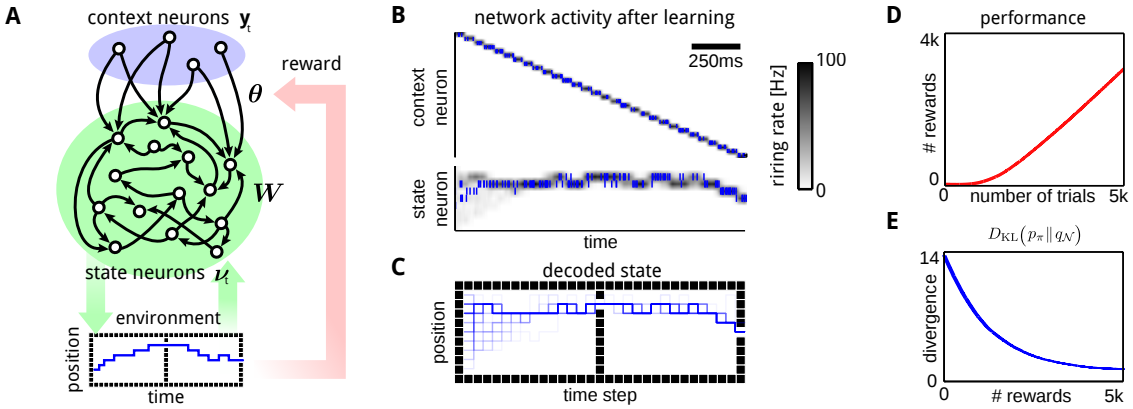

Figure 1. Illustration of the model for finite horizon planning. **A:** The neural network architecture considered here for solving the probabilistic planning problem. A recurrent layer of state neurons (green) that control the behavior of the agent receive feedforward input from context neurons (blue), the activity of which determine the desired goal. **B,C:** A simple planning problem that requires to pass through a passage at two specific points in time. Superposition of network activity averaged over 100 trial runs (B) and decoded network states (C) are shown. Blue dots in (B) show 1 example spike train. **D:** The accumulated number of rewards for the spiking network. **E:** The Kullback-Leibler divergence between the learned distribution and the true posterior. (C) and (D) show averages over 100 trial runs.

update rule.

We will compare *online* and *offline* updates of equation (7). In its stochastic *online* variant, the E-step is approximated by sampling a finite set of L samples to estimate the expectation,[24] or in the simplest case after a single sample ($L = 1$) as done in our experiments. We refer to this as the online approximation of equation (7). With *offline* updates, implemented as batch learning, the KL divergence $D_{KL}(p(\underline{\mathbf{v}}|r=1)||q(\underline{\mathbf{v}};\boldsymbol{\theta}))$ between the true posterior for planning in equation (1) and the model distribution converges to zero for $L \to \infty$ (assuming an exact encoding of the state and the transition model). This KL divergence establishes the relation between the inference problem for planning in equation (1) and the introduced problem of finding the network parameters that maximize the expected return.

A finite horizon planning task

To evaluate the spiking neural network model we consider a simple one dimensional planning problem, where the agent moves on a linear track and the activity of the state neurons population V directly determines its position. $K = 9$ state neurons encode nine discrete locations. A final reward is only received if the agent passes through two obstacles, one at time $T/2$ and one at time T (see Fig. 1C). Furthermore the agent is constrained not to jump to distant states within one time step. We model this constraint by the state transition model, i.e., $\mathscr{T}(\mathbf{v}_t = k \mid \mathbf{v}_{t-1} = i) = 1/3$, if $k-1 \leq i \leq k+1$ and (close to) zero otherwise (see the supplement for further details).

Due to the limitation on the state transitions, this problem requires planning ahead in order to avoid the obstacles successfully, i.e., to start moving to the passage before the obstacle actually appears. We show that the optimal planning policy can be learned using the reward modulated update rule in equation (7) in a network where the state neurons follow (soft) winner-take-all (WTA) dynamics. The probability $\rho_{t,k}$ of neuron k to spike at time t is given by $\rho_{t,k} = \exp(u_{t,k}) \,/\, \sum_{l=1}^{K} \exp(u_{t,l})$. Thus, in each

time step exactly one state neuron is active and encodes the current position of the agent.

The precise timing required to solve this task can be learned if the context neurons provide sufficient temporal structure. We study here the case of only one context neuron being active for one time-step, i.e., $y_{t,j} = 1$ for $j = t$ and $y_{t,j} = 0$ else. The weights θ_{kj} were adapted according to the online approximation of equation (7). Prior to learning, the agent performs a random walk according to the state transition model encoded in the weights w_{ki}, performing successful trials only occasionally. As learning proceeds, the activity of the context neurons shapes the behavior of the agent leading to nearly optimal performance. Figure 1D shows the accumulated reward throughout learning. After 5000 training iterations, the network generates rewarded trajectories in $97.80 \pm 4.64\%$ of the trials. We also evaluated a more detailed spiking version of the network model which produced similar results (success rate: $87.40 \pm 15.08\%$, see Fig. 1B,C in the supplement).

In addition to the online learning rule, we also evaluated the offline update rule. The network draws samples from a fixed distribution $q(\underline{\boldsymbol{\nu}}; \boldsymbol{\theta}_0 = 0)$ simulating random walks without any input (the initial state distribution $p(\boldsymbol{\nu}_0)$ was uniform). Offline updates are applied to the parameters $\boldsymbol{\theta}$ and Kullback-Leibler divergence converges towards zero with an increasing number of updates as shown in Fig. 1E.

Extension to the infinite horizon problem

Previously, we demonstrated how our network can model finite horizon planning tasks with terminal rewards (i.e., returns). Here we generalize to infinite horizon planning problems where rewards can be received at any point in time. The goal of the planning problem is to optimize the parameters $\boldsymbol{\theta}$ in the neural network so that it generates infinite trajectories that maximize the expected total discounted reward $\langle \sum_{t=0}^{\infty} \gamma^t \hat{r}_t \rangle$, where γ is the discount factor. We can reformulate this planning problem as probabilistic inference in an infinite mixture of Markov chains of finite lengths T.[1] The corresponding mixture distribution of trajectories is given by

$$q(r, \underline{\boldsymbol{\nu}}; \boldsymbol{\theta}) = \sum_{T=1}^{\infty} p(T)\, q(\underline{\boldsymbol{\nu}}_T; \boldsymbol{\theta})\, p(r|\underline{\boldsymbol{\nu}}_T) \tag{8}$$

where $p(T) = (1-\gamma)\gamma^T$ is the prior distribution over trajectory lengths, and $q(\underline{\boldsymbol{\nu}}_T; \boldsymbol{\theta})$ is the distribution over spike trains of length T according to the network dynamics in equation (4). The probability of getting a reward r at the end of the trajectory is given by $p(r = 1|\underline{\boldsymbol{\nu}}_T)$. Intuitively, in infinite horizon planning tasks the agent seeks a solution that balances getting to the goal as fast as possible (imposed by the prior) against the cost of large state jumps (imposed by the state transition model).

For the infinite horizon model we consider network dynamics where each state neuron has a sigmoid activation function, i.e., $\rho_{t,k} = \sigma(u_{t,k})$ with $\sigma(u) = 1/(1+e^{-u})$. Using this activation function we find that for learning the infinite planning task the parameters θ_{kj} should undergo a change in each time step t where the reward is present, according to

$$\Delta\theta_{t,kj} \;=\; \eta\, \hat{r}_t \sum_{\tau=1}^{t} \sum_{T=t-\tau}^{\infty} p(T)\, y_{\tau,j} \left(\nu_{\tau,k} - \rho_{\tau,k}\right) \;=\; \eta\, \hat{r}_t \sum_{\tau=1}^{t} \gamma^{t-\tau}\, y_{\tau,j} \left(\nu_{\tau,k} - \rho_{\tau,k}\right) \;. \tag{9}$$

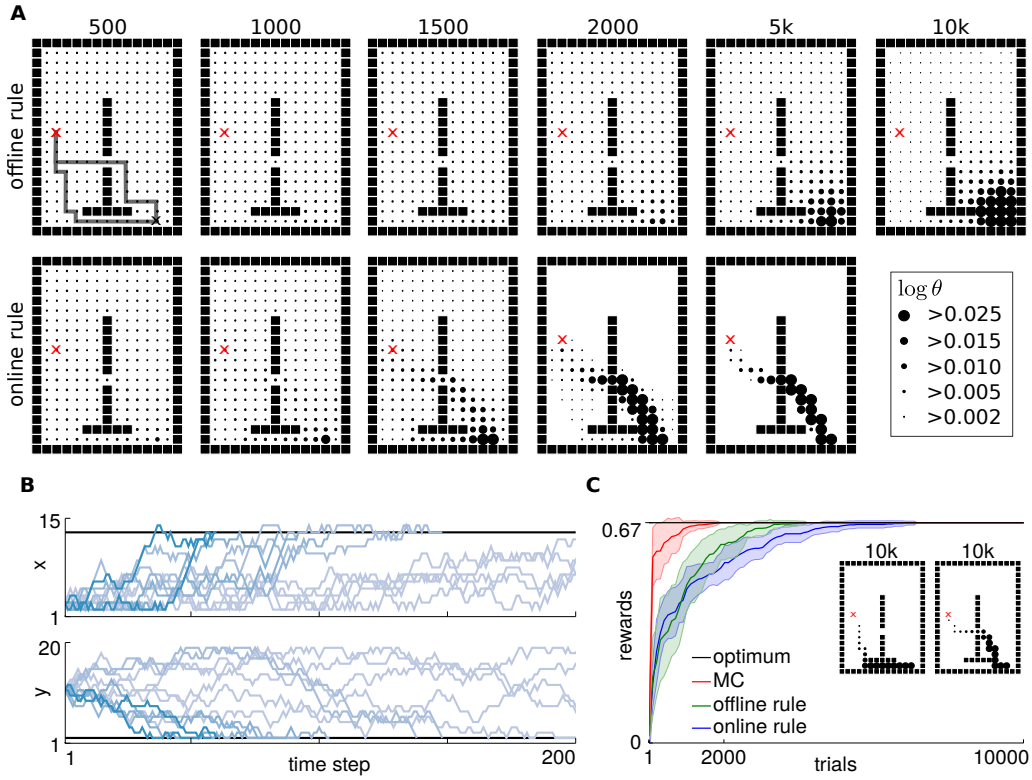

Figure 2. Illustration of the model for infinite horizon planning. **A:** The agent has to move from the red cross the black cross. The radii of the dots are proportional to the log of the $\boldsymbol{\theta}$ parameters. The results for the offline and the online learning rules are shown in the two rows, respectively. **B:** Illustration of 12 sampled trajectories after 10000 trials of offline learning. **C:** The mean of the received rewards over 20 experiments. We compare to Monte-Carlo policy evaluation (MC).

This synaptic weight update can be realized using an eligibility trace[?] $e_{t,kj}$ associated to each synapse with dynamics

$$e_{t,kj} = \gamma e_{t-1,kj} + y_{t,j}\left(v_{t,kj} - \rho_{t,kj}\right) \quad \text{and} \quad \theta_{t,kj} = \theta_{t-1,kj} + \eta\, r_t\, e_{t,kj}\,. \tag{10}$$

The eligibility trace is updated in each time step, whereas the weight updates are only applied for $\hat{r}_t > 0$. More details on the learning rule can be found in the supplement.

Note that the precise timing of attracting or repelling states cannot be modeled through explicit context neurons per time step as in the finite horizon model (since $t \to \infty$). Therefore, we consider stationary activity patterns of context neurons. This assumption implies that after convergence of the parameter updates an attractor cannot be visited twice.

An infinite horizon planning task

To test the infinite horizon model we consider a planning task, where the goal of the agent is to navigate from a given initial state to a target state in a grid maze with obstacles (dimensions $[15 \times 20]$). The network has 300 state neurons, one for each grid cell. The agent can perform only one-step moves, to the left, to the right, up or down, with equally probable transitions in each direction, and receives a reward $\hat{r}_t = 1$ only at the target state. The sampling process is either terminated if the target state

is reached, or if the time step exceeds the maximum number of allowed steps ($T = 300$). The discount factor was $\gamma = 0.98$.

With the offline learning rule the learned parameters $\boldsymbol{\theta}$ setup a gradient towards the target state, which covers *multiple* solution trajectories that lead to high total received rewards ($\boldsymbol{\theta}_0$ is chosen such that the agent starts at the initial state). This gradient is indicated by the radii of the dots in the first row in Fig. 2A. Figure 2B illustrates 12 example trajectories with weights obtained after 10000 trials of learning. Out of the shown 12 trajectories 9 reached the target state that is denoted by the black horizontal lines.

With the online learning rule, the learned parameters $\boldsymbol{\theta}$ specialize on one locally optimal path through the maze, which is illustrated in the second row of Fig. 2A. In the evaluated example, there are two locally optimal trajectories that are also the global optima. They are shown in the inset in Fig. 2C. For both, the offline and the online updates the average received reward converges to the maximum value (see Fig. 2C), where we compare to Monte-Carlo policy evaluation (MC).[20]

Results

A computational model for hippocampal sweeps

We show that the neural network reproduces the transient firing recorded in place cells in rats during planning phases. We compare our model predictions to recent results in,[9] where the authors analyzed the neural activity of 250 simultaneously recorded hippocampal neurons during phases of mental planning (using a 40-tetrode microdrive). In the experiments the animals received a reward, in an alternating manner, either at a known home location or at some (unknown) random location (one out of 36 locations arranged in a grid in a 2×2m maze). Here, we model only events that were observed while the animal was *resting* at some known current location and plans a route to the memorized home location (mental planning). An example event of a rat is illustrated in Fig. 3C, which illustrates the transient in the reconstructed position posterior probabilities. The resulting movement plan, i.e., the decoded and summed place cells' activity across time is shown in the lower panel in Fig. 3C.

In our network, the current location and the target location of the rat are modeled by $N = 20$ context neurons. The activity of which is denoted by $\mathbf{y}(t)$ and shown in Fig. 3A. The first ten context neurons encode the current location through a transient pattern. The remaining ten context neurons represent the desired target location. These neurons fire with a stationary Poisson process throughout the experiment. In this experiment we did not learn the weights of the context neurons. The weights were set proportional to the Euclidean distance of the preferred position of place cells to the initial state or the home location.

The network activity of 100 place cells (the preferred positions of which are aligned with a 10-by-10 grid) is solely driven by the context neurons and the recurrent weights. The recurrent synapses implement a Gaussian state transition model which prevents that the network directly draws samples close to the target state. In contrast to the ideal model used in the previous experiments multiple place cells can be active simultaneously, i.e., the recurrent weights encode a soft WTA circuit as in.[25]

Encoding continuous state variables

A simple encoding scheme is used to reconstruct the two dimensional state of the simulated rat $\boldsymbol{x}(t)$ from the place cells' activity, i.e., $\boldsymbol{x}(t) = 1 / |\boldsymbol{\nu}(t)| \sum_{l=1}^{L} (\nu_l(t)\, \boldsymbol{p}_l)$, where $|\boldsymbol{\nu}(t)| = \sum_{l=1}^{L} \nu_l(t)$ and $\boldsymbol{p}_l$ is the preferred position of place cell l.

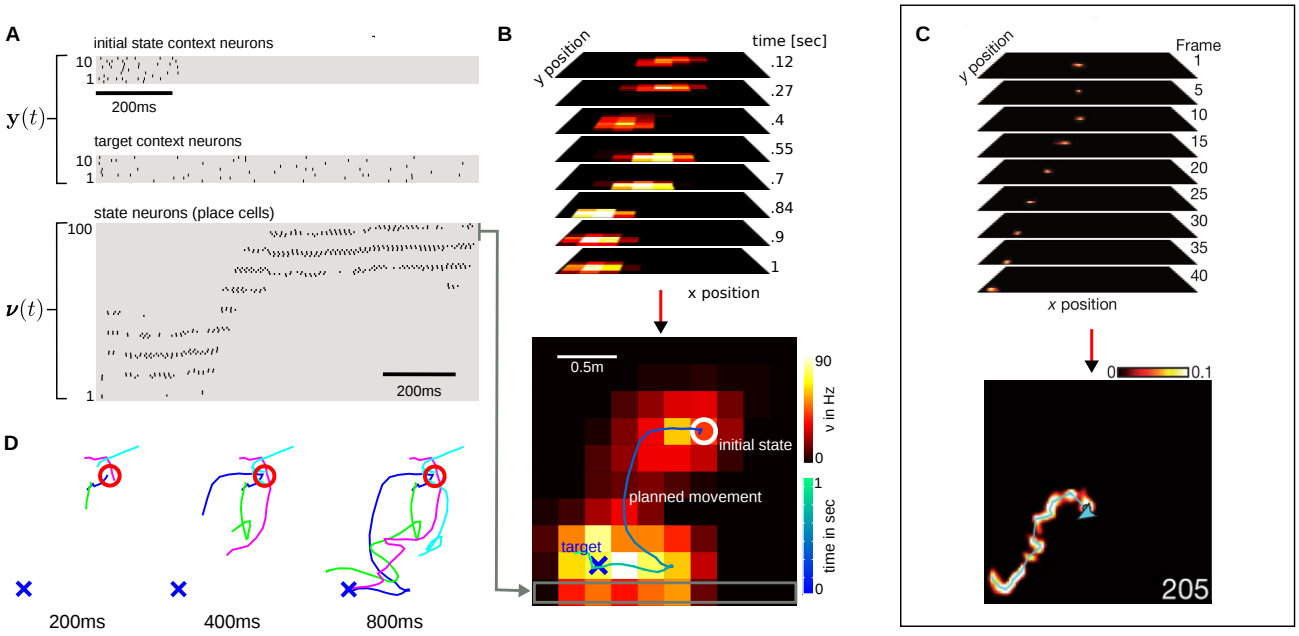

Figure 3. Planning and modeling transient firing in place cells with simulation results in (A-B,D). **C:** Biological data showing the position posterior probabilities in selected frames (the event duration was 205ms). Reprinted by permission from Macmillan Publishers Ltd: Nature 497, 74–79, © 2013. **A:** The first ten context neurons are only activated for the first 200ms to initialize the network at the desired initial state. The ten context neurons in the 2nd row in (A) implement a gradient towards the desired target state. The transient neural activity of 100 place cells is shown in the 3rd row in (A). The reconstructed movement trajectory is denoted by the line in (B). **D:** The result of four planning processes for an increasing planning horizon.

The generated transient activity of the place cells realize a path between the initial and the home location as shown in the 3rd row in Fig. 3A. The reconstructed movement plan is denoted by the line in Fig. 3B, where the integrated activity over the whole movement duration of each place cell is encoded by the color of the corresponding grid position. The simulated sequential firing shows a coherence to the transient firing in place cells in rats that is illustrated in Fig. 3C (see also the online material in a work of B. Pfeiffer and D. Foster[9]). A notable difference from the biological data is the resolution of the simulated activity in Fig. 3B. To visualize the corresponding spike events only 100 place cells were modeled. For a higher density of 900 uniformly distributed place cells we refer to additional results provided in the supplement.

Task adaptation through context neurons in a real robot

Typical robot planning tasks have to consider a large number of constraints that dynamically change and planning algorithms have to adapt to new solutions online. Here, we show that by activating context neurons multiple constraints can be modeled and used for task adaptation. As test platform we used a KUKA lightweight arm controlled in a two dimensional Cartesian space. The network generated movement plans in $2D$ and the complexity of the control problem itself is absorbed by a built-in Cartesian tracking controller. The two coordinates modeled, x and y span the transverse plane. The trajectory was executed using inverse kinematics to obtain a reference joint trajectory and inverse dynamics control to execute it.

We used $K = 225$ state neurons that receive excitatory input from context neurons modeling the initial state and the target state (as in the previous experiment), see Fig. 4A. Strong inhibitory input is used to model obstacles (the gray areas in Fig. 4B).

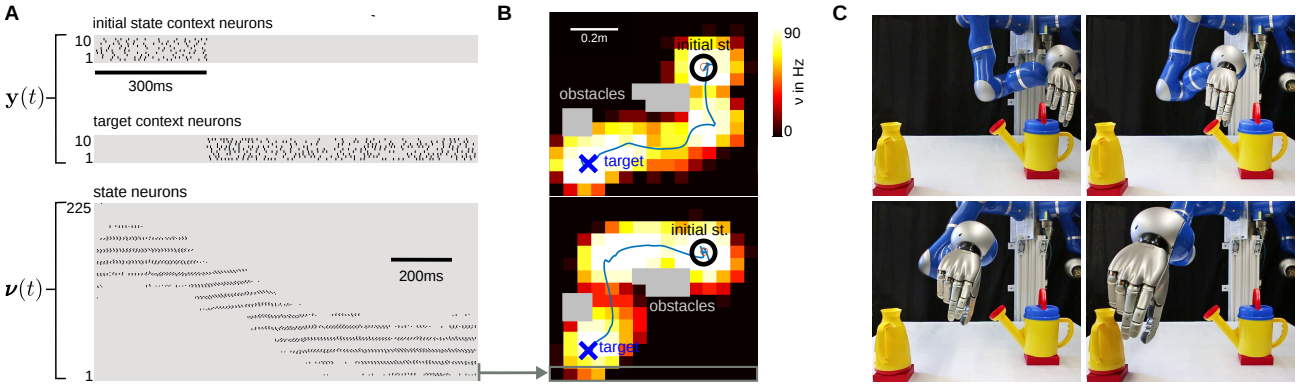

Figure 4. Planning with multiple constraints on a real robot. **A:** Generated spike train (top: context neurons, bottom: state neurons) after contrastive divergence learning of the transition model. **B:** Two sampled movement plans solving the obstacle avoidance task. **C:** Snapshots of the executed movement on the real robot.

The transition model was learned from demonstrated state transitions (through kinesthetic teaching) in contrastive divergence learning[21] (see the supplement for details).

After learning, the network generates goal-directed movement plans that can be used for obstacle avoidance. Sampling a movement plan of a duration of 1.3 seconds took on average 635 ± 8 milliseconds on a standard computer (the symbol $\pm$ denotes the standard deviation computed from 1000 trajectories). The minimum distance to the target was 2.71 ± 2.24 cm in an operation area of 70×70 cm.

An interesting property of the spiking model is that multiple solutions can be encoded in the same model. For the considered obstacle avoidance task, two solutions are shown in Fig. 4B and snapshots of the second solution are depicted in Fig. 4C.

Discussion

The brain efficiently processes and predicts sequential activity patterns in the context of planning tasks.[9–12] Understanding these processes and how information is possibly encoded in stochastic neurons are major goals of theoretical neuroscience. In this paper, we demonstrated how recurrent spiking neural networks can solve planning problems and provide a solid theory based on the framework of probabilistic inference. The model reproduces observed neural dynamics, predicts that contextual information is one of the key modulating factors and is a promising low-energy control strategy for mobile robot applications.

Theoretical contributions

Spiking neural networks are a reasonable neuroscience model as verified by data.[26–29] Their capabilities in solving planning tasks however have been underexplored to date. It was shown that spiking networks can encode and draw samples from arbitrary complex distributions,[30,31] models of temporal sequences can be learned[7,8] and Bayesian filtering was studied.[29,32–34]

Our model builds on these probabilistic sampling results and a solution to planning problems is suggested that approximates a stochastic process for planning through forward sampling from a parameterized model distribution. The parameters of the model distribution denote the synaptic weights of a population of afferent neurons for which local Hebbian update rules were

derived from the principle of probabilistic inference. The derivations include arbitrary differentiable activation functions and postsynaptic potential shapes for the neuron model (details are provided in the supplement).

Links to expectation maximization[22,24] and policy gradient methods[?,?,?] were established, where the resulting offline learning rules are similar to Monte Carlo policy evaluation[20] and the network parameters subject to these updates converge to the globally optimal solution for WTA network dynamics. The online learning rules resemble the online Monte Carlo policy iteration algorithm and converge to local lower bounds of the optimum.

The correctness of the neural planning method was validated in two toy tasks, a finite horizon planning task with a known optimal solution and an infinite horizon task, where the neural network achieved the same performance level as Monte-Carlo policy evaluation.[20]

Implications for neuroscience

Previous neural models that implement path planning have focused on attractor networks or potential fields[14,35–38] and their activity was related to hippocampal firing.[16,17] A deficit of these models is however that the path which was taken to reach a desired state cannot be modeled with attractors. To overcome this limitation a sequence of successive metastable states in attractor networks was proposed[18] but it is left unclear how these networks can be trained from rewards. We followed a different approach where attractors emerge through reinforcement of rewarding trajectories. As a result, different input neurons with its synaptic weights can model different routes to multiple attractors. Thus, the proposed model extends the modulation abilities of attractor networks and can be validated, e.g., in a study on contrasting planning of safe versus straight-line paths.

We demonstrated in simulation results that the proposed recurrent neural network can reproduce the dynamically changing firing rates observed during hippocampal sweeps.[9] The input modulated activity in our network hypothesizes that a cognitive map representation (the recurrently connected state neurons) receives contextual input from other brain regions. Potential sites for these contextual inputs are projections from the entorhinal and the prefrontal cortex.[39,40] It is worth mentioning that the network is not limited to model hippocampal sweeps. It may be used to model frequently observed dynamically changing firing rates from other brain regions.[41–44]

Embedded in the framework of probabilistic inference, the proposed network can be naturally extended in multiple ways, e.g., the place cells encoding the state transition model might be learned,[16] actions might be encoded additionally,[25] forward and backward replays[10,11] can be simulated, or multiple cognitive maps can be installed.[17] Furthermore, Poisson neurons were chosen for simplicity and the model generalizes to noisy integrate and fire neurons.[45]

Implications for robotics

State-of-the-art planning algorithms in robotics generate movement plans within seconds and scale to many degrees of freedom.[?,?,?] Spiking neural networks could not compete with these methods due to the encoding of continuous variables, e.g., in our robot experiment, we used population codes[?,?] to encode a two-dimensional continuous state variable and more than a few hundred neurons could not be simulated without risking to run out of memory in a standard computer. A potential

solution that is currently under investigation are factorized population codes which scale but can not capture correlations (which are needed to avoid obstacles). Another promising alternative are neuromorphic chips which were already used to learn 62-dimensional joint angle sequences of human jump motions in recurrent networks.[?] In addition, related spiking network models were proposed which also build on winner-take-all circuits and local plasticity rules.[?,?,?] Therefore, it is reasonable to assume that the presented theory provides the basis for future neural controller implementations in neuromorphic hardware for robot control.

In contrast to previous work on spiking neurons in a reinforcement learning framework,[25] we followed here a model-based approach where the recurrent dynamics of the network can be reused to learn multiple related tasks with different sets of weights from the context neurons (e.g., representing different goal positions or obstacles). The state transition model does not need to be re-learned when switching between environments.

Furthermore, our model has the advantage that multi-modal solutions to planning tasks can be learned. This feature was exploited in an obstacle avoidance task in a real robot, where the network randomly sampled one out of two paths. This ability to encode non-linear mappings is in particular beneficial for learning forward and inverse kinematic models in robotics.

References

1. Toussaint, M. & Storkey, A. Probabilistic inference for solving discrete and continuous state markov decision processes. In *proceedings of the ICML*, 945–952 (ACM, 2006).

2. Kappen, H. J., Gómez, V. & Opper, M. Optimal control as a graphical model inference problem. *Machine Learning* **87**, 159–182 (2012).

3. Botvinick, M. & Toussaint, M. Planning as inference. *Trends in Cognitive Sciences* **16**, 485 – 488 (2012).

4. Solway, A. & Botvinick, M. M. Goal-directed decision making as probabilistic inference: A computational framework and potential neural correlates. *Psychological Review* **119**, 120–154 (2012).

5. Penny, W. D., Zeidman, P. & Burgess, N. Forward and backward inference in spatial cognition. *PLoS Comput Biol* **9** (2013).

6. Pezzulo, G., van der Meer, M. A., Lansink, C. S. & Pennartz, C. M. Internally generated sequences in learning and executing goal-directed behavior. *Trends in cognitive sciences* **18**, 647–657 (2014).

7. Brea, J., Senn, W. & Pfister, J.-P. Sequence learning with hidden units in spiking neural networks. In *Advances in Neural Information Processing Systems*, 1422–1430 (2011).

8. Kappel, D., Nessler, B. & Maass, W. STDP installs in winner-take-all circuits an online approximation to hidden Markov model learning. *PLoS Comp. Biol.* **10**, e1003511 (2014).

9. Pfeiffer, B. & Foster, D. Hippocampal place-cell sequences depict future paths to remembered goals. *Nature* **497**, 74–79 (2013).

10. Foster, D. J. & Wilson, M. A. Reverse replay of behavioural sequences in hippocampal place cells during the awake state. *Nature* **440**, 680–683 (2006).

11. Johnson, A. & Redish, A. D. Neural ensembles in ca3 transiently encode paths forward of the animal at a decision point. *The Journal of neuroscience* **27**, 12176–12189 (2007).

12. Carr, M. F., Jadhav, S. P. & Frank, L. M. Hippocampal replay in the awake state: a potential substrate for memory consolidation and retrieval. *Nature Neuroscience* **14**, 147–153 (2011).

13. Hopfield, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences* **79**, 2554–2558 (1982).

14. Samsonovich, A. & McNaughton, B. L. Path integration and cognitive mapping in a continuous attractor neural network model. *The Journal of neuroscience* **17**, 5900–5920 (1997).

15. McNaughton, B. L., Battaglia, F. P., Jensen, O., Moser, E. I. & Moser, M.-B. Path integration and the neural basis of the'cognitive map'. *Nature Reviews Neuroscience* **7**, 663–678 (2006).

16. Erdem, U. M. & Hasselmo, M. A goal-directed spatial navigation model using forward trajectory planning based on grid cells. *European Journal of Neuroscience* **35**, 916–931 (2012).

17. Azizi, A. H., Wiskott, L. & Cheng, S. A computational model for preplay in the hippocampus. *Frontiers in computational neuroscience* **7** (2013).

18. Rabinovich, M., Huerta, R. & Laurent, G. Transient dynamics for neural processing. *Science* **321**, 48–50 (2008).

19. Rawlik, K., Toussaint, M. & Vijayakumar, S. On stochastic optimal control and reinforcement learning by approximate inference (extended abstract). In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI'13, 3052–3056 (AAAI Press, 2013).

20. Sutton, R. S. & Barto, A. G. *Reinforcement learning: An introduction*, vol. 28 (MIT press, 1998).

21. Hinton, G. Training products of experts by minimizing contrastive divergence. *Neural computation* **14**, 1771–1800 (2002).

22. Dempster, A. P., Laird, N. M. & Rubin, D. B. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)* **39**, 1–38 (1977).

23. Todorov, E. Linearly-solvable markov decision problems. In *NIPS*, 1369–1376 (2006).

24. Wei, G. C. G. & Tanner, M. A. A monte carlo implementation of the em algorithm and the poor man's data augmentation algorithms. *Journal of the American Statistical Association* **85**, 699–704 (1990). http://www.tandfonline.com/doi/pdf/10.1080/01621459.1990.10474930.

25. Frémaux, N., Sprekeler, H. & Gerstner, W. Reinforcement learning using a continuous time actor-critic framework with spiking neurons. *PLoS Comp. Biol.* **9**, e1003024 (2013).

26. Gerstner, W. & Kistler, W. M. *Spiking neuron models: Single neurons, populations, plasticity* (Cambridge university press, 2002).

27. Izhikevich, E. M. *et al.* Simple model of spiking neurons. *IEEE Transactions on neural networks* **14**, 1569–1572 (2003).

28. Izhikevich, E. M. Which model to use for cortical spiking neurons? *IEEE transactions on neural networks* **15**, 1063–1070 (2004).

29. Deneve, S. Bayesian spiking neurons i: inference. *Neural computation* **20**, 91–117 (2008).

30. Buesing, L., Bill, J., Nessler, B. & Maass, W. Neural dynamics as sampling: A model for stochastic computation in recurrent networks of spiking neurons. *PLoS Comput Biol* **7**, e1002211 (2011).

31. Berkes, P., Orbán, G., Lengyel, M. & Fiser, J. Spontaneous cortical activity reveals hallmarks of an optimal internal model of the environment. *Science* **331**, 83–87 (2011).

32. Bobrowski, O., Meir, R. & Eldar, Y. C. Bayesian filtering in spiking neural networks: Noise, adaptation, and multisensory integration. *Neural computation* **21**, 1277–1320 (2009).

33. Boerlin, M. & Denève, S. Spike-based population coding and working memory. *PLoS computational biology* **7**, e1001080 (2011).

34. Legenstein, R. & Maass, W. Ensembles of spiking neurons with noise support optimal probabilistic inference in a dynamically changing environment. *PLoS computational biology* **10**, e1003859 (2014).

35. Glasius, R., Komoda, A. & Gielen, S. C. Neural network dynamics for path planning and obstacle avoidance. *Neural Networks* **8**, 125–133 (1995).

36. Miller, W. T., Werbos, P. J. & Sutton, R. S. *Neural networks for control* (MIT press, 1995).

37. Stringer, S., Rolls, E., Trappenberg, T. & De Araujo, I. Self-organizing continuous attractor networks and path integration: two-dimensional models of place cells. *Network: Computation in Neural Systems* **13**, 429–446 (2002).

38. Lebedev, D. V., Steil, J. J. & Ritter, H. J. The dynamic wave expansion neural network model for robot motion planning in time-varying environments. *Neural Networks* **18**, 267–285 (2005).

39. Keefe, J. O. & Nadel, L. *The hippocampus as a cognitive map* (Clarendon Press Oxford, 1978).

40. Redish, A. D. *Beyond the cognitive map: from place cells to episodic memory* (MIT Press Cambridge, MA, 1999).

41. Abeles, M. *et al.* Cortical activity flips among quasi-stationary states. *Proceedings of the National Academy of Sciences* **92**, 8616–8620 (1995).

42. Jones, L. M., Fontanini, A., Sadacca, B. F., Miller, P. & Katz, D. B. Natural stimuli evoke dynamic sequences of states in sensory cortical ensembles. *Proceedings of the National Academy of Sciences* **104**, 18772–18777 (2007).

43. Luczak, A., Barthó, P., Marguet, S. L., Buzsáki, G. & Harris, K. D. Sequential structure of neocortical spontaneous activity in vivo. *Proceedings of the National Academy of Sciences* **104**, 347–352 (2007).

44. Zhang, Q.-f. *et al.* Priming with real motion biases visual cortical response to bistable apparent motion. *Proceedings of the National Academy of Sciences* **109**, 20691–20696 (2012).

45. Rao, R. P. Hierarchical bayesian inference in networks of spiking neurons. In Saul, L., Weiss, Y. & Bottou, L. (eds.) *Advances in Neural Information Processing Systems 17*, 1113–1120 (MIT Press, 2005).

Acknowledgements

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreements No. 248311 (AMARSI) and No. 600716 (CoDyCo). The authors would like to thank Wolfgang Maass, Matthew Botvinick and Tucker Hermans for comments that greatly improved the manuscript. We would also like to thank Brad Pfeiffer and David Foster for the permission to print parts of their inspiring results.[9]

Author contributions

E.R., D.K. and D.P. conceived the story, designed the model and derived the update rules. E.R. and D.K. performed the model simulations. D.T. implemented contrastive divergence learning. D.T. and E.R. performed the real robot experiment. E.R., D.K., D.P., and J.P. wrote the manuscript.

Additional information

Competing financial interests: The authors declare no competing financial interests.

Bibliography

[1] A. Paraschos, E. Rueckert, J Peters, and G. Neumann. Model-free probabilistic movement primitives for physical interaction. 2015.

[2] E. Rueckert, D. Kappel, D. Tanneberg, D Pecevski, and J. Peters. Recurrent spiking networks solve planning tasks. *Scientific Reports*, 2016.

[3] E. Rueckert, M. Mindt, J. Peters, and G. Neumann. Robust policy updates for stochastic optimal control. 2014.