Deep Spiking Networks for Model-based Planning in Humanoids

Daniel Tanneberg[1], Alexandros Paraschos[1], Jan Peters[1,2] and Elmar Rueckert[1]

***Abstract*— We propose a novel bioinspired motion planning approach based on deep networks. This Deep Spiking Network (DSN) architecture couples task and joint space planning through bidirectional feedback. We show that the DSN can learn arbitrary complex functions, encode forward and inverse models, generate different solutions simultaneously and adapt dynamically to changing task constraints or environments. Furthermore, to scale to high-dimensional spaces, we introduce a factorized population coding in the model. Moreover, we show that the DSN can be trained efficiently and exclusively from human demonstrations to learn a task independent and re-usable planning model. The model is evaluated in simulation and on two real high-dimensional humanoid robotic systems.**

I. Introduction

Autonomous robots are expected to assemble parts in industry, assist in critical surgeries, help in households, or work in environments which are dangerous for humans. A fundamental skill that is required in almost all robotic tasks is planning, i.e., the computation of movement trajectories which accomplish the given task while fulfilling certain constraints. For example, collision-free arm trajectories for parts assembly need to be planned. Most planning algorithms assume known constraints or features and known models that map these constraints into the configuration space of the robot [1], [2], [3], [4]. However, in the general case, these models are unknown and need to be learned from data.

Recent developments in deep learning approaches tackled model learning and developed *end-to-end* learning approaches to directly learn control policies from sensory input streams [5], [6], [7], [8], [9]. However, an open question is how deep networks could re-use parts of the learned model to generate movement plans to new environments or tasks without re-learning. Therefore, we propose a recurrent spiking neural network (SNN) that is able to learn a task independent and re-usable planning model from human demonstrations. By using stochastic spiking neurons in the network, it is related to neural processes and exploration is an intrinsic neuronal function that reduces the amount of required training data. Spiking networks can encode arbitrary complex distributions [10] and learn temporal sequences [11], [12]. We utilize these properties to learn forward and inverse kinematics as well as to encode multi-modal trajectory distributions that can represent multiple solutions to planning problems. Constraints and task-related information can be integrated into the model by appropriate excitatory or inhibitory input resulting in a flexible and adapting architecture. Multiple different solutions can be sampled efficiently and in parallel. This is important to avoid collisions and react to changing environments. Particularly in a robotic co-worker scenario, this can be used to avoid dangerous situations for humans.

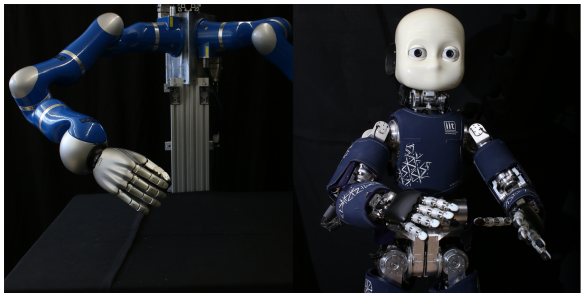

Fig. 1. *(left)* a KUKA lightweight arm where we used 6 DoFs and *(right)* an iCub where we used the right arm with 7 DoFs. The proposed DSN is able to learn task independent models for both complex systems and to plan movements in task and joint space simultaneously.

The proposed bioinspired Deep Spiking Network is based on a task space planning approach, that was used to model maze navigation in rats [13]. In this approach, a population of recurrently connected state neurons was used to encode a transition model. The activity of these state neurons is modulated by contextual neurons injecting desired initial and target state information. Connections between state and contextual neurons are learned through a local reward modulated Hebbian learning rule. To model the operational space, a full population coding approach was used which does not scale to more than three dimensions. We propose a Deep Spiking Network using a factorized population coding to scale to high-dimensional systems and focus on learning the task-independent model from human demonstrations gathered from complex real systems.

The contribution of this work is a bioinspired hierarchical neural network architecture and corresponding learning rules to train the model efficiently and exclusively from data. We added a layer of factorized joint space populations to a recent two-dimensional task space model to construct the proposed Deep Spiking Network that scales to high dimensions. The DSN couples the task and joint space of the robot via bidirectional feedback and generates smooth task and joint space trajectories simultaneously. In [14] it was shown, that such interchanging feedback supports the generation of smooth trajectories. No kinematic and state transition models are required as we show that the non-linear

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 #610967 (TACMAN).

[1]Intelligent Autonomous Systems, Technische Universität Darmstadt, 64289 Darmstadt, Germany
{tanneberg,paraschos,rueckert}@ias.tu-darmstadt.de
[2]Robot Learning Group, Max-Planck Institute for Intelligent Systems, 72076 Tübingen, Germany, mail@jan-peters.net

forward and inverse kinematics are learned together with the state transition models. The learned model is evaluated in simulation and on real robotic systems showing its abilities to dynamically adapt to changing environments, producing smooth high-dimensional trajectories and finding multiple solutions at once.

II. Movement Planning with Spiking Neural Networks

We build our Deep Spiking Network on top of the two-dimensional task space model from [13]. To scale to higher-dimensional joint space, we introduce a three-layered network and derive novel efficient learning rules to train the model from spatio-temporal human demonstrations.

A. Two-dimensional task space model

The model consists of two different kinds of neuron populations, one of K *state neurons* and one of N *context neurons*. State neurons encode the operational space and context neurons encode the task, e.g., desired goals or obstacles. Each constraint or any task-related information is modeled by a population of context neurons. While the state neurons are uniformly spaced within the modeled state space, the context neurons are Gaussian distributed around the corresponding location they encode. There are no lateral connections between context neurons, but each context neuron $j \in N$ connects to all state neurons $k \in K$ by a synaptic weight $\theta_{j,k}$. The state neuron population is fully connected by synaptic weights $w_{i,k}$, connecting state neuron i to state neuron k.

We denote the activity of the state neurons by $\mathbf{v}_t = (v_{t,1}, .., v_{t,K})$, where $v_{t,k} = 1$ if state neuron k spiked at time t and $v_{t,k} = 0$ otherwise. The activity of the context neurons is denoted the same way by $\mathbf{y}_t$. The synaptic weights $w_{i,k}$ encode the state transition model $T(\mathbf{v}_t|\mathbf{v}_{t-1})$ that encodes how likely the state transitions are. All neurons have a preferred position in a Cartesian coordinate system and encode binary random variables (spike/no spike). Thus, the solution to a planning problem is the spike train of the state neurons, i.e., a sequence of binary activity vectors $\mathbf{v}_t$. To get a movement trajectory, the spike train needs to be decoded into a sequence of continuous spatial locations. This decoding of the state $\mathbf{x}_t$ (e.g., position in task or joint space) from neural activity is done by a simple decoding scheme [15], [16], $\mathbf{x}_t = \frac{1}{|\hat{\mathbf{v}}_t|}\sum_{k=1}^{K}\hat{v}_{t,k}\mathbf{p}_k$ with $|\hat{\mathbf{v}}_t| = \sum_{k=1}^{K}\hat{v}_{t,k}$, where $\mathbf{p}_k$ denotes the preferred position of neuron k and $\hat{v}_{t,k}$ is its Gaussian window filtered neural activity at time t. With this decoding, continuous states can be encoded by binary activity patterns.

The state neurons can be seen as an abstract and simplified version of place cells and encode a cognitive map of the environment. They are modeled by stochastic neurons which build up a membrane potential based on the weighted neural input. Context neurons have no afferent connections and spike with a fixed time-dependent probability. Operating in discrete time and using a fixed refractory period τ that decays linearly, the neurons spike in each time step with a probability based on their membrane potential. All spikes from presynaptic neurons get weighted by the corresponding synaptic weight and are integrated to an overall postsynaptic potential (PSP). Assuming linear dendritic dynamics, the membrane potential of the state neurons is given by

$$u_{t,k} = \sum_{i=1}^{K} w_{i,k}\tilde{v}_i(t) + \sum_{j=1}^{N}\theta_{j,k}\tilde{y}_j(t) \ , \tag{1}$$

where $u_{t,k}$ denotes the membrane potential for neuron k at time t, $\tilde{v}_i(t)$ and $\tilde{y}_j(t)$ denote the effects of PSPs from state neuron i and context neuron j respectively. Thus, Equation (1) defines a simple stochastic version of the spike response model [17]. Using this membrane potential, the probability to spike for the state neurons can be defined by $\rho_{t,k} = p(v_{t,k} = 1) = f(u_{t,k})$, where $f(\cdot)$ denotes the activation function, that is required to be differentiable. Now the probability for generating a state sequence $\mathbf{v}_{1:T}$ of length T starting from a given initial state $\mathbf{v}_0$ is defined by the model distribution

$$\begin{aligned} q(\mathbf{v}_{1:T}|\boldsymbol{\theta}) &= p(\mathbf{v}_0)\prod_{t=1}^{T}\prod_{k=1}^{K}\rho_{t,k}^{v_{t,k}}(1-\rho_{t,k})^{1-v_{t,k}} \\ &= p(\mathbf{v}_0)\prod_{t=1}^{T}T(\mathbf{v}_t|\mathbf{v}_{t-1})\phi_t(\mathbf{v}_t;\boldsymbol{\theta}) \ , \end{aligned} \tag{2}$$

where $\phi(\cdot)$ denotes the task-specific input. The chosen activation function is an exponential with $f(\cdot) = \exp(\cdot)$. In this work we aim to learn a task-independent model, i.e., we are only interested in the recurrent connections of the state neurons. Thus, in contrast to [13] where the synaptic weights $\theta_{j,k}$ between context and state neurons are learned through reinforcement learning, we use handcrafted weights. The weights are set automatically for each task using the Euclidean distance between the modeled constraint and the preferred location of the neuron. The neural activity $\mathbf{y}_t$ of the context neurons is determined by a probability depending on the kind of constraints they encode. As the context neurons encode task-related information, they are designed and activated for solving a particular task. The transition model is task-independent and used in the unconstrained stochastic process that defines a freely moving agent as defined in Equation (2) without ϕ_t. Sampling from this distribution can be implemented by a recurrent SNN [11], [12], [18], which generates random walk trajectories of length T. Injected task-related input modulates these random walk trajectories towards goal-directed and constraint-fulfilling movement plans.

B. Context neurons for (online) task adaption

In typical planning problems, the planner has to consider a number of constraints that may even change dynamically. The architecture of the SNN implements a simple task adaption mechanism. Context neurons with proper synaptic weights and activity patterns have to be added or changed, i.e., choosing θ and $\mathbf{y}_t$, to adapt to changes. Targets are modeled by context neurons with excitatory connections to the state neurons and an activity pattern depending on the

time the target should be reached. Obstacles are modeled by context neurons with inhibitory connections. Therefore, neurons covered by an obstacle are deactivated and no sampled movement can cross this area. By adapting the synaptic connections or the activity patterns online, dynamic changes in the environment or the task occurring during planning can be taken into account.

C. Deep Spiking Network model for coupling the dimensions

As envisioned in the beginning, we want to get joint space planning, where the joint space of robots is usually high-dimensional such that the full population code from [13] will not work in practice. Thus, to scale to higher dimensions, we propose a factorized population code approximation using F independent one-dimensional models. Each factorized model is built in the aforementioned fashion and encodes one joint. Using this factorized approximation makes SNNs feasible for planning in higher dimensional spaces in general. However, as the factorized models are independent and cannot exchange information during planning, they cannot deal with complex scenarios, e.g., like obstacles in the environment.

A possible solution to overcome the missing ability of the factorized independent models to deal with obstacles, is to combine a lower dimensional model for planning in task space, that can deal with obstacles, with F independent joint space models. These consist of M_f state neurons each, that encode the higher dimensional joint space and the bidirectional mapping. In robotics this mapping is known as the *kinematics* and defines a mapping from positions in task space (e.g. Cartesian x, y coordinates of an endeffector) to an appropriate joint configuration (inverse kinematics) or vice versa (forward kinematics). Therefore, we propose a Deep Spiking Network (Figure 2(A)) that is built by combining the two-dimensional task space planner with factorized joint space and kinematics models. This model can solve planning problems in task space and transforms the movement into joint angle trajectories simultaneously. The state neurons of the factorized models encode the joint space and the state neurons of the task space model span the operational space of the robot. State neurons in each space are fully connected and project to all state neurons in the other space through symmetric weights ψ. These connections enable the model to project feedback from task to joint space and vice versa. It has been shown that such links facilitate smooth trajectories [14].

The synaptic weights ψ encode forward and inverse kinematics at once. Adding feedback from joint to task space during planning alters the membrane potential in Equation (1) of the task space state neurons to

$$u_{t,k} = \sum_{i=1}^{K} w_{i,k}\tilde{v}_i(t) + \sum_{j=1}^{N} \theta_{j,k}\tilde{y}_j(t) + \sum_{l=1}^{M} \psi_{l,k}\tilde{z}_l(t) \ , \quad (3)$$

where $\psi_{l,k}$ denotes the synaptic weight between joint space neuron l and task space neuron k, $\tilde{z}_l(t)$ denotes the PSPs from joint space state neuron l and $M = \sum_{f=1}^{F} M_f$ is the total number of joint space state neurons. The membrane potential of the joint space state neurons is defined analogously

$$o_{t,k} = \sum_{j=1}^{M} \delta_{j,k}\tilde{z}_j(t) + \sum_{i=1}^{K} \psi_{k,i}\tilde{v}_i(t) \ , \quad (4)$$

where $o_{t,k}$ is the potential of neuron k at time t and $\delta_{j,k}$ is its afferent connection with neuron j. Adding the factorized models and the bidirectional feedback, the model distribution from Eq. (2) expands to

$$q(\mathbf{v}_{1:T}, \mathbf{z}_{1:T}|\boldsymbol{\theta}) = p(\mathbf{v}_0)p(\mathbf{z}_0)\prod_{t=1}^{T}\varsigma(t) \quad \text{with} \quad (5)$$
$$\varsigma(t) = T(\mathbf{v}_t|\mathbf{v}_{t-1}, \mathbf{z}_{t-1})T_{\mathrm{K}}(\mathbf{z}_t|\mathbf{v}_{t-1}, \mathbf{z}_{t-1})\phi_t(\mathbf{v}_t; \boldsymbol{\theta}) \ ,$$

where $\mathbf{z}$ denotes the binary activity vector of the joint space state neurons defined like $\mathbf{v}_t$ and $T_{\mathrm{K}}(\mathbf{z}_t|\mathbf{v}_{t-1}, \mathbf{z}_{t-1}) = \prod_{f=1}^{F} T_{\mathrm{K}_f}(\mathbf{z}_{t,f}|\mathbf{v}_{t-1}, \mathbf{z}_{t-1,f})$ is the factorized transition model of the joint space taking feedback from task space into account. Figure 2(B) shows the graphical model representation of the proposed Deep Spiking Network, highlighting the bidirectional feedback and time dependencies.

D. Learning of the Deep Spiking Network

For learning the transition models from human demonstrations, we use maximum likelihood learning from single spike samples. In particular, we use contrastive divergence (CD) [19] which approximates the maximum likelihood gradient by drawing samples from a proposed distribution. In [19] it was suggested that only a few Markov chain Monte Carlo (MCMC) cycles are required to obtain useful approximations of the expectations in practice and that even a single MCMC cycle has sufficient information to let the learning process converge to the maximum-likelihood solution. Thus, the CD update equation for parameter Θ of function $f(x; \Theta)$ is given by

$$\Delta\Theta = \left\langle \frac{\partial \log f(x;\Theta)}{\partial\Theta} \right\rangle_{\mathbf{X}^0} - \left\langle \frac{\partial \log f(x;\Theta)}{\partial\Theta} \right\rangle_{\mathbf{X}^1} , \quad (6)$$

where $\mathbf{X}^0$ and $\mathbf{X}^1$ denote the state of the Markov chain after 0 and 1 cycles respectively. We chose CD as it fits to our online learning scenario of replaying human demonstrations, takes samples from the current model into account and provides efficient learning rules.

We want to learn the transition models encoded in the synaptic weights between the state neurons using human demonstrations recorded from the robot through kinesthetic teaching to get a task-independent model according to the area covered by the robot. Therefore, we derive a spike dependent version of contrastive divergence to learn these synaptic weights, similar to results in [20]. The transition model of our DSN model is given from Equation (5) without task input, i.e., no $\phi_t(\mathbf{v}_t; \theta)$, and consists of the two parts

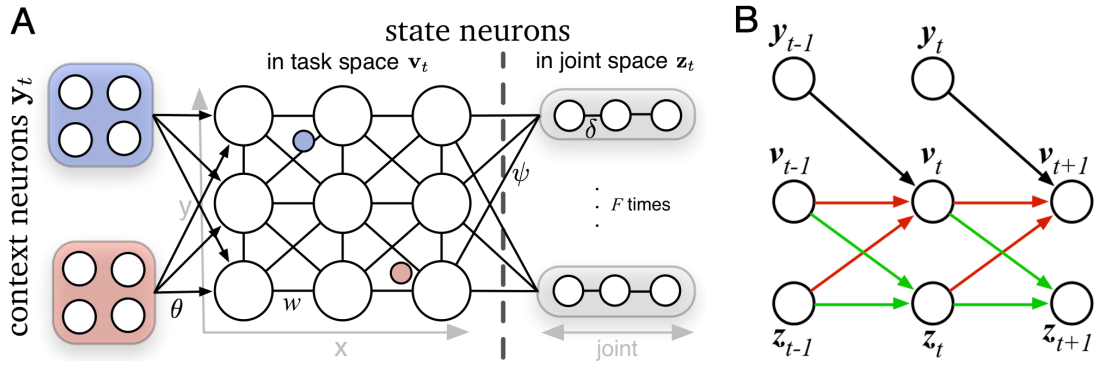

Fig. 2. (A) Sketch of the proposed DSN, showing the two-dimensional task space model (w) on the left, the F independent joint space models (δ) on the right and the kinematics models (ψ) between. Not all synaptic connections are shown to keep the sketch clear. The blue and the red dot indicate two task space constraints (e.g., initial and target position) encoded by the corresponding context neurons. (B) Graphical model representation of the proposed DSN , where $\mathbf{v}_t$ and $\mathbf{z}_t$ are sampled separately in each time step t, indicated by the colored connections. During learning there is no task layer $\mathbf{y}_t$.

$T(\mathbf{v}_t|\mathbf{v}_{t-1}, \mathbf{z}_{t-1})$ and $T_{\mathrm{K}}(\mathbf{z}_t|\mathbf{v}_{t-1}, \mathbf{z}_{t-1})$ that are given by

$$\exp\left(\sum_{k=1}^{K} w_{k,i} v_{t-1,k} v_{t,i} + \sum_{m=1}^{M} \psi_{m,i} z_{t-1,m} v_{t,i}\right) \quad \text{and}$$

$$\exp\left(\sum_{k=1}^{M_f} \delta_{k,m} z_{t-1,f,k} z_{t,f,m} + \sum_{i=1}^{K} \psi_{m,i} v_{t-1,i} z_{t,f,m}\right)$$

respectively and f indicates the dimension. Here we consider a resetting rectangular PSP kernel of one time step ($v_{t-1,k}$). Using a PSP kernel of τ time steps ($\tilde{v}_k(t)$) follows the same derivation and is used in the experiments to be more realistic and to take information provided by spikes from multiple previous time steps into account. We start by differentiating the logarithm of the task space transition model ($\log T(v_{t,i}|\mathbf{v}_{t-1}, \mathbf{z}_{t-1})$) w.r.t. $w_{k,i}$, leading to

$$\frac{\partial \log T(v_{t,i}|\mathbf{v}_{t-1}, \mathbf{z}_{t-1})}{\partial w_{k,i}} = v_{t-1,k} v_{t,i}.$$

Analogously we differentiate $\log T(v_{t,i}|\mathbf{v}_{t-1}, \mathbf{z}_{t-1})$ and $\log T_{\mathrm{K}_f}(z_{t,f,m}|\mathbf{v}_{t-1}, \mathbf{z}_{t-1})$ for $\psi_{m,i}$ and $\delta_{k,m}$. Inserting those derivatives into Equation (6) leads to the following update rule for $w_{k,i}$

$$\Delta w_{k,i} = \langle v_{t-1,k} v_{t,i} \rangle_{\mathbf{X}^0} - \langle v_{t-1,k} v_{t,i} \rangle_{\mathbf{X}^1} \ , \qquad (7)$$

and analogous to similar update rules for $\psi_{m,i}$ and $\delta_{k,m}$. Learning is done by replaying the human demonstrations as spike trains to the model while updating the synaptic weights. The demonstrations include arbitrary movements covering the operational space of the robot and were recorded using kinesthetic teaching. At each time step t there is only one training sample for each synaptic weight in form of activity pairs $(\tilde{v}_{t-1,k}, \tilde{v}_{t,i})$. The training spike pattern, denoted by $\tilde{v}$, is compared with the sample drawn from the current model, denoted by v, using the presynaptic training data as input. Using a single sample for each space in each time step, this leads to the final synaptic update rules

$$\Delta w_{k,i} = \tilde{v}_{t-1,k} \tilde{v}_{t,i} - \tilde{v}_{t-1,k} v_{t,i} \qquad \text{and} \qquad (8)$$

$$\Delta \psi_{m,i} = \tilde{z}_{t-1,m} \tilde{v}_{t,i} - \tilde{z}_{t-1,m} v_{t,i} \qquad (9)$$

for the task space transition model and the forward kinematic models and to

$$\Delta \delta_{k,m} = \tilde{z}_{t-1,k} \tilde{z}_{t,m} - \tilde{z}_{t-1,k} z_{t,m} \qquad \text{and} \qquad (10)$$

$$\Delta \psi_{m,i} = \tilde{v}_{t-1,i} \tilde{z}_{t,m} - \tilde{v}_{t-1,i} z_{t,m} \qquad (11)$$

for the joint space transition models and the inverse kinematic models. In each time step the samples $v_{t,i}$ and $z_{t,m}$ are drawn alternatively with the presynaptic input $\mathbf{v}_{t-1}$ and $\mathbf{z}_{t-1}$ as depicted in Figure 2(B) without task input $\mathbf{y}_{t-1}$, as there is no particular task during learning. As long as the samples drawn from the current model do not match the training data, the model keeps updating its synaptic weights.

Equations (8)-(11) imply that state neurons which spike successively will increase their synaptic weights and those that do not will decrease their synaptic weights. Using this learning scheme, we can learn the state transition models in task space (Eq. (8)) and in joint space (Eq. (9)), as well as the forward (Eq. (10)) and inverse (Eq. (11)) kinematics mappings simultaneously, resulting in an efficient way for learning the proposed DSN from demonstrations.

III. Results

In the experiments, we learned models from human demonstrations from the robotic systems shown in Figure 1 and solved obstacle avoidance tasks with these models. For a statistical comparison, we defined a two-dimensional obstacle avoidance setup with changing initial and target positions and compared our model to the model from [13] in simulation. Additionally, we evaluated the model's ability to adapt to changes in the environment during planning by adding a simulated dynamic obstacle, i.e., an obstacle that moves with unknown dynamics during planning.

A. Learning of the DSN

Using kinesthetic teaching, we recorded 15 minutes of arbitrary movements sampled at 1 ms covering the operational space of the KUKA arm. Joint angle trajectories of the six joints and the x, y position of the endeffector were recorded. Task space and joint space trajectories were transformed into spike trains using Gaussian basis functions centered at the state neurons and inhomogeneous Poisson processes. All

synaptic weights were initialized as inhibitory connections. In total learning of the complete DSN took 58 minutes on a standard desktop computer using MATLAB. More details on the network can be found in the Appendix.

TABLE I

EVALUATION OF THE MODELS ON OBSTACLE AVOIDANCE TASKS WITHIN 70×70CM ON THE KUKA ARM IN SIMULATION.

	2D model [13]	**DSN** (T)	**DSN** (J)
Acceptance rate	0.97	0.95	0.56
Comp. time (ms)	300 ± 25	954 ± 176	954 ± 176
Avg. jerk (cm/s^3)	0.88 ± 0.17	0.92 ± 0.16	0.91 ± 0.17
Target error (cm)	1.64 ± 1.39	1.56 ± 1.18	3.59 ± 1.90

B. Model comparison

To evaluate the learned model, we defined an obstacle avoidance setup with 10 different combinations of initial and target positions and evaluated 100 trajectories for each setup and model. Note that the model does not need to be retrained for the different tasks, i.e., the model can adapt to changing tasks like different via-points or obstacles. The computational time is measured as the time required to sample and decode a movement trajectory in task *and* joint space. The average jerk of the trajectories is calculated in time steps of 20 ms and serves as a smoothness criteria. The target error denotes the smallest Euclidean distance of the trajectories to the desired target averaged with a Gaussian window filter using a time window of 20 ms. An error of 1 cm within the modeled area of 70×70 cm corresponds to a relative error of about 1%. The acceptance rate is calculated as the ratio of accepted samples to total samples generated. Samples are rejected if their target error > 5 cm, their average jerk > 0.12 cm/s^3 or if they collide with an obstacle. As the Deep Spiking Network generates a movement trajectory in task space (T) and simultaneously the corresponding trajectories in joint space (J), we evaluated both results. However, the DSN does not need kinematic models neither for learning nor for executing the joint space trajectories. Only for calculating the criteria of the joint angle trajectories, forward kinematics were used for mapping into task space. Table I shows the results of the two-dimensional task space model and our DSN on obstacle avoidance tasks in simulation. Hyperparameters of both models were optimized using stochastic search [21] with acceptance rate as objective.

The task space trajectories of the two-dimensional task space model and of the DSN achieve equally good results on the target error (around 1.60 cm) and acceptance rate (around 96%). The joint space trajectories of the DSN achieve a target error of 3.59 cm and have a lower acceptance rate of 56%. This may reflect inaccuracies in the learned kinematic models and the only task-related information in joint space is provided through the feedback from this mapping. Although the acceptance rate of the joint angle trajectories is lower, the accepted trajectories are equally smooth and achieve a similar precision. Thus, the proposed Deep Spiking Network can solve a task space planning problem and as additional result provides joint angle trajectories that can be executed on the robot without post processing and the need of a task space controller or the inverse kinematics of the robot. We validated that by executing trajectories sampled from the DSN for an execution time of 4 s on the real robot. Figure 3 shows the model's ability to simultaneously encode multiple solutions to a given task as both solutions can be sampled from the model in parallel.

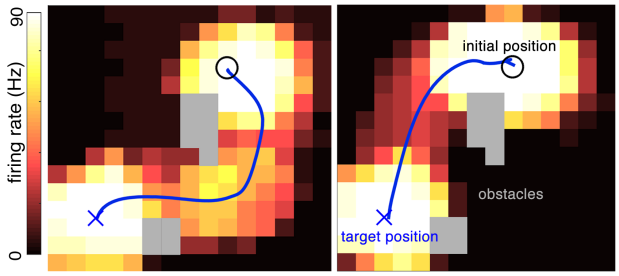

Fig. 3. The activity of the task space neurons of two different sampled solutions shown as heatmaps and the decoded movement trajectories on top.

C. Adapting to moving obstacles

To test the model's ability to dynamically adapt to changes in the environment or task during planning, we added a dynamic obstacle in addition to a static obstacle. The obstacle movement was initialized after 300 ms and moved 40 cm within 300 ms. The proposed model was able to generate successful trajectories of equal quality. Multiple solutions were found, covering different strategies of avoiding that simulated dynamic obstacle, e.g., walking around it or waiting until the obstacle has passed by. Figure 4(A-B) shows a sampled solution of the task with a static and a dynamic obstacle.

D. Movement planning on a humanoid robot

To evaluate the proposed model on an additional complex system, we gathered training data of the right arm of the iCub robot in the same way as described before. In total we recorded 15 minutes of data sampled at 10 ms from the seven joints and the hand as endeffector covering an operational space of 15×30 cm. After learning the model, we evaluated it on a set of obstacle avoidance tasks in simulation and executed sampled joint angle trajectories on the Gazebo iCub simulation. The obtained results match the results on the KUKA arm shown in Table I. Namely, acceptance rates of 0.93 and 0.79, average jerks of 0.0019 ± 0.0004 cm/s^3 and 0.0030 ± 0.0006 cm/s^3 and target errors of 0.0071 ± 0.0040 cm and 0.0105 ± 0.0057 cm for the task and joint space trajectories respectively. Note that the covered operational area of the iCub is much smaller and the movements are more constrained compared to the KUKA experiments.

IV. CONCLUSION

In this work, we demonstrated that SNNs are suitable models to solve robot planning tasks and come with useful features like learning arbitrary complex functions. In particular, the proposed DSN learns forward and inverse models and can adapt dynamically to changing environments and constraints. We introduced a Deep Spiking Network that uses factorized population coding to scale to high-dimensional problems and couples task and joint space during planning

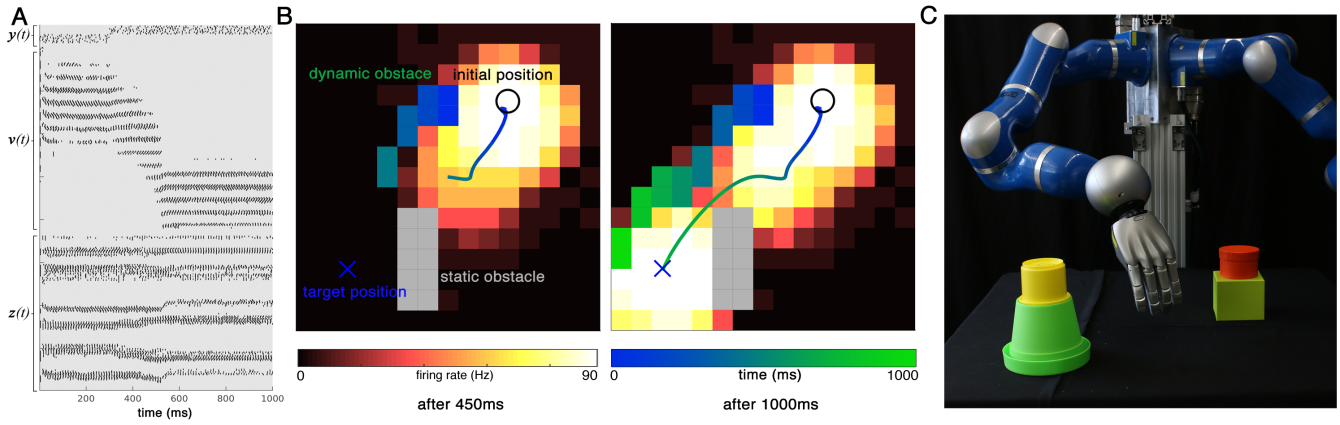

Fig. 4. (A) Spike train of a sampled movement for the dynamic obstacle task. (B) The activity of the task space state neurons ($\mathbf{v}_t$) of (A) is shown as a heatmap with the decoded movement trajectory on top. Grey blocks indicate static obstacles, colored blocks represent the moving obstacle. The color of the movement trajectory and of the moving obstacle encode time. The colors never match, showing that the sampled movement trajectory successfully avoids the moving obstacle. (C) Illustration of the experiment setup with two static obstacles and the KUKA arm.

and learning. It can generate multiple plans for a task in parallel, enabling foresighted robot control by providing alternatives to choose from. We derived novel contrastive divergence learning rules for efficient training from human demonstrations. On two robotic platforms, we showed that it is feasible to use the proposed DSN on complex real systems.

As we aimed for learning a task-independent model, we kept the properties of the context neurons fixed. These can be learned through reinforcement learning but need to be relearned for every task [13]. Additionally, the current model only uses feedforward signals, or assumes a low-level tracking controller. For future work, feedback should alter the feedforward signals, resulting in model-predictive control with SNNs and robotic constraints like torque or joint limits need to be considered during planning.

APPENDIX

Network details: Initial and target positions are modeled with 10 context neurons each. The timing of their spikes is defined by the kind of position they encode, i.e., the population that encodes the initial position is active within the first 300 ms. This allows the network to initialize at the given initial position. After this time, these neurons are deactivated and the neurons encoding the target position are active until the predefined sample length ($T = 1000$, simulating 1 s) is reached. Obstacles can be modeled by strong inhibition on the covered state neurons, implemented through an offset term. To model the covered task spaces of the robots, we use 15 task space state neurons for each dimension. Each of the factorized models consists of 30 state neurons to model the joint space. All state neurons were spaced within the dimension limits $[-1, +1]$ and movements were mapped accordingly. The refractory period and the PSP window are set for all neurons to $\tau = 10$ ms.

REFERENCES

[1] S. Lavalle and J. Kuffner, "Rapidly-Exploring Random Trees: Progress and prospects," in *Algorithmic and Computational Robotics: New Directions*, 2000.

[2] L. Kavraki, P. Švestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, 1996.

[3] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "CHOMP: Gradient optimization techniques for efficient motion planning," in *Int. Conf. Robotics and Automation*, 2009.

[4] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "STOMP: Stochastic trajectory optimization for motion planning," in *Int. Conf. Robotics and Automation*, 2011.

[5] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *J. Mach. Learn. Res.*, 2016.

[6] S. Lange and M. Riedmiller, "Deep auto-encoder neural networks in reinforcement learning," in *Int. Joint Conf. on Neural Networks*, 2010.

[7] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," *The International Journal of Robotics Research*, 2015.

[8] K. Noda, H. Arie, Y. Suga, and T. Ogata, "Multimodal integration learning of robot behavior using deep neural networks," *Robotics and Autonomous Systems*, 2014.

[9] L. Pinto and A. Gupta, "Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours," *arXiv*, 2015.

[10] L. Buesing, J. Bill, B. Nessler, and W. Maass, "Neural dynamics as sampling: A model for stochastic computation in recurrent networks of spiking neurons," *PLoS Computational Biology*, 2011.

[11] J. Brea, W. Senn, and J. Pfister, "Sequence learning with hidden units in spiking neural networks," in *Advances in Neural Information Processing Systems*, 2011.

[12] D. Kappel, B. Nessler, and W. Maass, "STDP installs in winner-take-all circuits an online approximation to hidden markov model learning," *PLoS Comput. Biol*, 2014.

[13] E. Rueckert, D. Kappel, D. Tanneberg, D. Pecevski, and J. Peters, "Recurrent spiking networks solve planning tasks," *Nature PG: Scientific Reports*, 2016.

[14] M. Toussaint and C. Goerick, "A Bayesian view on motor control and planning," in *From Motor Learning to Interaction Learning in Robots*, Springer, 2010.

[15] A. Georgopoulos, A. Schwartz, and R. Kettner, "Neuronal population coding of movement direction," *Science*, 1986.

[16] W. Ma, J. Beck, P. Latham, and A. Pouget, "Bayesian inference with probabilistic population codes," *Nature neuroscience*, 2006.

[17] W. Gerstner and W. Kistler, *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press, 2002.

[18] Y. Huang and R. Rao, "Neurons as monte carlo samplers: Bayesian inference and learning in spiking networks," in *Advances in Neural Information Processing Systems*, 2014.

[19] G. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural Comput.*, 2002.

[20] E. Neftci, S. Das, B. Pedroni, K. Kreutz-Delgado, and G. Cauwenberghs, "Event-driven contrastive divergence for spiking neuromorphic systems," *Frontiers in Neuroscience*, 2014.

[21] N. Hansen, S. Müller, and P. Koumoutsakos, "Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES)," *Evolutionary computation*, 2003.