Robotics: Science and Systems 2019
Freiburg im Breisgau, June 22-26, 2019

Conditional Neural Movement Primitives

M. Yunus Seker*, Mert Imre*, Justus Piater† and Emre Ugur*
*Computer Engineering Department Bogazici University, Istanbul, Turkey
Email: {yunus.seker1, mert.imre, emre.ugur}@boun.edu.tr
†Department of Computer Science, Universität Innsbruck, Austria
Email: justus.piater@uibk.ac.at

Abstract—**Conditional Neural Movement Primitives (CNMPs) is a learning from demonstration framework that is designed as a robotic movement learning and generation system built on top of a recent deep neural architecture, namely Conditional Neural Processes (CNPs). Based on CNPs, CNMPs extract the prior knowledge directly from the training data by sampling observations from it, and uses it to predict a conditional distribution over any other target points. CNMPs specifically learns complex temporal multi-modal sensorimotor relations in connection with external parameters and goals; produces movement trajectories in joint or task space; and executes these trajectories through a high-level feedback control loop. Conditioned with an external goal that is encoded in the sensorimotor space of the robot, predicted sensorimotor trajectory that is expected to be observed during the successful execution of the task is generated by the CNMP, and the corresponding motor commands are executed. In order to detect and react to unexpected events during action execution, CNMP is further conditioned with the actual sensor readings in each time-step. Through simulations and real robot experiments, we showed that CNMPs can learn the non-linear relations between low-dimensional parameter spaces and complex movement trajectories from few demonstrations; and they can also model the associations between high-dimensional sensorimotor spaces and complex motions using large number of demonstrations. The experiments further showed that even the task parameters were not explicitly provided to the system, the robot could learn their influence by associating the learned sensorimotor representations with the movement trajectories. The robot, for example, learned the influence of object weights and shapes through exploiting its sensorimotor space that includes proprioception and force measurements; and be able to change the movement trajectory on the fly when one of these factors were changed through external intervention.**

I. Introduction

Acquiring an advanced robotic skill set requires a robot to learn complex temporal multi-modal sensorimotor relations in connection with external parameters and goals. Learning from demonstration (LfD) framework [1] has been proposed in robotics as an efficient and intuitive way to teach such skills to the robots, in which the robot observes, learns and reproduces the observed demonstrations. During teaching the skills, how the task is influenced by different factors is generally obvious to humans, yet this knowledge is mostly hidden in the experienced sensorimotor data of the robot and difficult to extract autonomously. Development of feature extraction and learning methods that are sufficiently general and flexible for a broad range of robotic tasks still stands as an important challenge in robotics. In order to deal with large variety of tasks that are influenced by factors defined in different levels of abstractions, we require a single framework that can learn feature-movement and raw-data-movement associations from small and large number of demonstrations, respectively, automatically filtering out the irrelevant information.

Another challenge in LfD is to deal with multiple trajectories: Multiple demonstrations might be required to teach a skill either because different action trajectories are required in different situations or simply because there are multiple means to achieve the same task even in the same settings. The robot, in return, needs to capture the important characteristics in these observations coming from several demonstrations, and be able to reproduce the learned skill in new configurations reacting to unexpected events, such as external perturbations, on the fly. For this, the robot needs to develop an understanding on if and how the sensorimotor data and externally provided parameters are related to each other and to the motor commands.

While one or more of the above properties were addressed by the existing movement frameworks [14, 15, 6, 2, 16, 4, 13], none of these approaches can handle all these requirements in one framework. In this paper, we propose Conditional Neural Movement Primitives (CNMP), a robotic framework that is built on top a recent neural network architecture, namely conditional neural processes (CNP), to encode movement primitives with the identified functionalities. Given multiple demonstrations, CNMP encodes the statistical regularities by learning about the distributions of the observations from reasonably few input data. Conditioning mechanism is used to predict a sensorimotor trajectory given external goals and the current sensorimotor information. Conditioning can be applied over the learned sensorimotor distribution using any set of variables such as joint positions, visual features or haptic feedback at any time point. For example for a learned grasp skill, the system might be queried to predict and generate hand and finger motion trajectory with a conditioning on the color of the object, weight measured in the wrist joint, and target aperture width for the finger. Given low- or high-dimensional input, CNMP can utilize simple MLPs or convolution operation, respectively, to encode the correlations. Such networks allow the system to automatically extract the features that are relevant to the task. Finally, the predicted trajectory is realized by generating the corresponding actuator commands. CNMP can also learn multiple modes of operations from multiple demonstrations of a movement primitive. Importantly, CNMP specifically produces movement trajectories in joint or task space, and generates the corresponding motor commands to achieve these trajectories through a high-level feedback control loop. The encoded sensorimotor representations are continuously conditioned through the feedback coming from the sensory readings, enabling the system to ensure match between predicted and actual trajectories and to respond to unexpected events in its sensorimotor space.

II. Related Work

LfD [1] has been applied to various robotic learning problems including object grasping and manipulation [6, 2, 16, 4, 13]. Among others, learning methods that are based on dynamic systems [19] and statistical modeling [5] have been popular in the recent years. Dynamic Movement Primitives (DMPs)[19] encode the demonstrated trajectory as a set of differential equations, and offers advantages such as one-shot learning of non-linear movements, real-time stability and robustness under perturbations with guarantees in reaching the goal state, generalization of the movement for different goals, and linear combination of parameters. The parameters of the system can be learned with different advanced algorithms such as Locally Weighted Regression (LWR) [3] and Locally Weighted Projection Regression [22]. Statistical modeling techniques, on the other hand, use probabilistic models such as Gaussian Mixture Models[6] or Hidden Markov Models[12] to encode the motion from multiple demonstrations where the statistical regularities are learned in the spatiatemporal data.

Zhou et al.[23] used a simple 2D obstacle avoidance task to show that standard DMP approach is not designed to learn from multiple trajectories and therefore cannot encode the important parts of multiple demonstrations. They developed a method that performs LWR by defining a new objective function dependent on the environment restrictions and parameters, and that uses a model-switching algorithm to be able to train over all the parametric space. Ude et al.[20] used LWR and GPR methods to compute new DMP based trajectories given parameters that describe the characteristics of the action, such as the goal point. Their method could be generalized with complex trajectories for both periodic and discrete movements. Pervez et al. [18] learned a separate GMM for each demonstration and mixed the separately learned GMMs to generate trajectories for new task parameter values.

In order to enable physical collaboration, the robot is required to learn haptic feedback models that encodes the default predicted force feedback measured during execution of the corresponding skill. Memorized force and tactile profiles have already been successfully utilized in modulating learned DMPs in difficult manipulation tasks that contain high degrees of noise in perception such as grasping and in-hand manipulation of objects from incorrect positions or flipping boxes using chopsticks [17]. HMMs were used to learn multi-modal models from temperature, pressure and fingertip information for exploratory object classification tasks [7]; and PHMMs were used to learn haptic feedback trajectory models to adapt actions in response to external perturbations [10, 21]. Deep networks [8] were used to learn multi-modal models from different sensory information such as temperature, pressure, fingertip, contacts, proprioception, and speech; however these models were used only to categorize the sensory data without any effect on action execution. More recently, the same problem generalization of force/torque profiles was investigated for contact tasks [11] where these profiles were modeled by Locally Weighted Regression (LWR) which has local generalization capabilities, hence successful at the intermediate query points.

III. Method

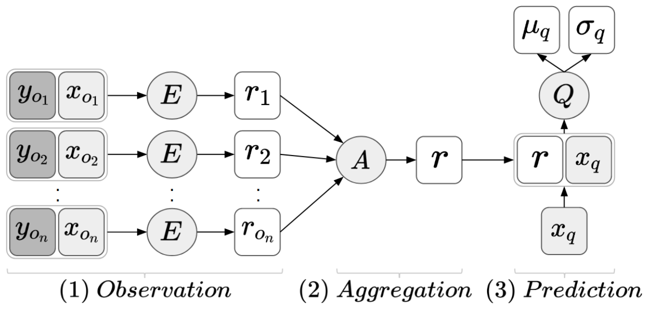


Fig. 1: Structure of the CNP adapted from [9].

A. Background

Until recently, most of the neural networks in deep learning were trained to approximate a single output function. However, when the data is a distribution which can be represented by statistical models such as Gaussian distributions, networks that approximates to the single functions cannot represent the underlying model. Instead the network can be modelled as a probabilistic approximator that can predict the distribution parameters, namely mean and variance, instead of producing one single output value. Garnelo et al. [9] proposed a new family of neural models that processes the training data based on Bayesian Inference principles by combining the advantages of the neural networks with Gaussian Processes. The proposed method is able to extract prior knowledge directly from the training data by sampling observations from it, and use this information to predict a conditional distribution over any other target points. The general structure of the model is shown in the Figure 1. The model consists of three parts: (1) A parameter-sharing Encoder Network E is used to encode all sampled observations into fixed size representations r_i in a latent space. Notice that the numbers and orders of the coming observations can change during both training and test time because empirical observations can change dynamically. (2) The individual representations are aggregated into one general representation r that covers all the observations sampled at the beginning. Therefore, according to the prior knowledge r, the model can predict a distribution over the target input/inputs by conditioning on the observations. (3) Finally, a Query Network Q produces distribution parameters (μ_q, σ_q) over the query input x_q by using the general representation r as a prior. As a summary, the CNP model can be formulated as

$$\mu_q, \sigma_q = Q_\theta \left(x_q \oplus \frac{\sum_i^n E_\phi((x_i, y_i))}{n} \right)$$

where $\{(x_i, y_i)\}_{i=0}^n$ are observation pairs sampled from data, E_ϕ is the encoder network with parameters ϕ, x_q is the target

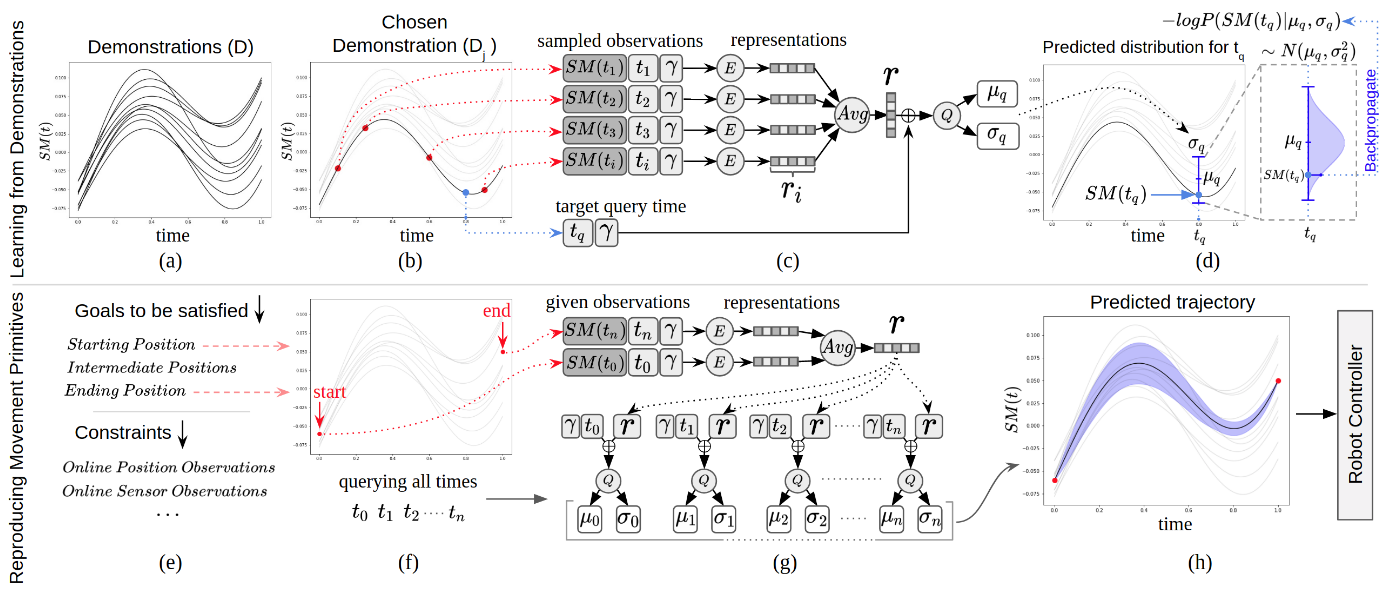


Fig. 2: Training and test procedures of Conditional Neural Movement Primitives. Refer to Section III for more details.

input, i.e., the query over which we want the model to predict a Gaussian distribution, $\oplus$ is the concatenation operator, and Q_θ is the query network with parameters θ that outputs the Gaussian distribution parameters (μ_q, σ_q). The whole model is trained together as a stochastic process where at each training iteration, n random observation pairs $(x_i, y_i)_{i=0}^{n\in[1,n_{\max}]}$ and one query point (x_q, y_q) are sampled from the training data. $n_{\max}$ is the hyperparameter giving the maximum number of observations that can be used during the training process. At the end of each iteration, neural network parameters (θ and ϕ) of both Encoder and Query network are optimized according to the following loss function:

$$\mathcal{L}(\theta, \phi) = -\log P(y_q \mid \mu_q, \text{softmax}(\sigma_q)) \qquad (1)$$

where μ_q and σ_q are the outputs of the CNP, y_q is the corresponding output of the x_q for this training iteration, and P is the Gaussian probability function that returns the conditional probability of the y_q for given mean and variance parameters.

B. Proposed Method: Conditional Neural Movement Primitives

Conditional Neural Movement Primitives (CNMP) is a learning from demonstration framework that is designed as a robotic movement learning and generation system built on top of CNPs. It specifically produces movement trajectories in joint or task space, and executes these trajectories through a high-level feedback control loop. The encoded sensorimotor representations are conditioned externally to constrain the movement and set goals. Additionally, as detailed later, CNMP is also continuously conditioned by the current sensory readings, predicting the desired sensorimotor values based on these readings. This allows to detect unexpected events and respond to these events by changing the movement trajectory on the fly.

A CNMP is trained for each skill that is taught to the robot using one or more demonstrations in possibly different configurations. In each demonstration, the robot stores the actuator positions, the perceptual features, and the corresponding time points. Therefore each demonstration D_j is stored as a trajectory of sensorimotor features and the corresponding time points, i.e. $\{(t, SM(t))\}_{t=0}^{t=\tau}$ where $SM(t)$ corresponds to the observed sensorimotor data at time t. SM can encode any information including task and joint space positions of the robot arm and gripper, force/torque measurements, low-dimensional environment properties such as size of manipulated objects or high-dimensional world state such as top-down 2D raw image mask of the environment. In the rest of this section, SM is described as a one dimensional vector which might be flexibly referred as sensory or motor data in different times for simplicity and clarity.

CNMP training and trajectory reproduction steps are explained in Fig. 2 and detailed in the next subsections. Recall that although SM is displayed as a one dimensional vector, it is normally a multi-dimensional vector that includes any sensory or motor information. Therefore, CNMP is designed to enable learning from and constraining in any set of dimensions in the SM space.

Learning from Demonstrations using CNMP: For each demonstration, first, time invariance is ensured by scaling the time values into [0,1]. In each training iteration, a demonstration D_j is selected randomly from a uniform distribution. From the chosen demonstration, n observation tuples $\{(t_i, SM(t_i))\}_{i=0}^{n} \in D_j$ and one target query $(t_q, SM(t_q)) \in D_j$ are sampled uniformly. n is a random number between $[0, n_{max}]$ where n_{max} is a hyper-parameter that sets the maximum number of samples collected from the chosen demonstration by CNMP. This allows our framework to be flexible over changing numbers of observations at the test time. Fig. 2.1c illustrates the neural network operations for

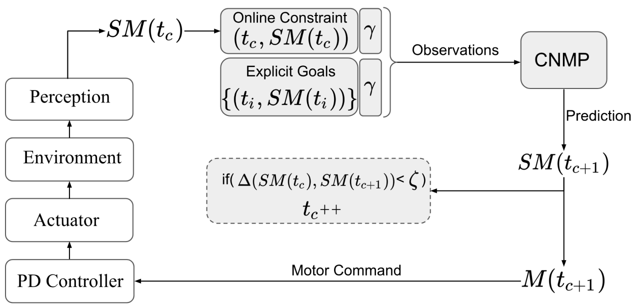


Fig. 3: CNMP execution loop with feedback

an example of four observation points. Each observation tuple goes through the parameter sharing Encoder Network (E) and produces the representation for the corresponding observations in a fixed sized latent space. Using a mean operator, individual representations are merged into one general representation r. The general representation r and target query time t_q are concatenated and given to the Query Network (Q) as input. As the final step of the neural network, Query Network predicts the distribution parameters (μ_q, σ_q) of SM for target time t_q. Figure 2.1d shows the predicted Gaussian distribution and the actual $SM(t_q)$ of the robot at the queried time. The error between the predicted and actual SM values is calculated using the loss function in Eq. 1 for each query. The computed errors are back-propagated to train the weights of the network. Task specific external parameters (γ) can also be included in encoding the movement primitive by directly concatenating with the observation and query data and providing to the Encoder and Query Networks directly. Such external parameters can be regarded as part of sensory data, therefore for simplicity we will not provide separate explanation for handling these parameters in the rest of this section.

Reproducing movement primitives: After learning from demonstrations, the CNMP is used to generate the SM trajectory that is predicted to satisfy the externally provided goals given the current sensorimotor information, where both goals and sensorimotor information are represented with a set of observation tuples. The external goals are set to the desired sensorimotor values at the desired time points ($\{(t_i, SM(t_i))\}$). The means of identifying and querying the goals is extremely flexible (2.2e). For example, the goals can refer to the starting, intermediate or final positions of the movement trajectory, and set once to generate and execute the desired trajectory. As another example, the goals can be set as the desired force/torque readings expected to be observed during movement generation. The current sensorimotor information, on the other hand, corresponds to the SM at the current time point (t_c): ($\{(t_c, SM(t_c))\}$). Continuous querying with the current SM enables the system to detect and respond to unpredicted events such as external perturbations that generates discrepancy between the actual and predicted sensorimotor values. In case such prediction error is detected, the movement trajectory changes on the fly, and executed as detailed in the next subsection. This effectively corresponds to autonomously reacting to external perturbations exploiting the encoded multi-modal distributions[1].

Figure 2.2f illustrates an example case where the goal is set as two test time observations that correspond to the desired start and end positions of the movement. These goal observations are given to the trained CNMP model and the general representation r is produced after the Encoding and Averaging operations in the neural network (Fig 2.2.g). This general representation r holds the compact information for all the given goal observations, thus, guides the Query network to produce movement primitive distributions that satisfy all the desired goals/constraints at the once. To extract the whole movement trajectory, all time steps ${t_i}_{i=0}^{1}$ are concatenated with the pre-computed general representation r, and the set of mean and variance values for the whole trajectory is predicted by using the Query network. Figure 2.2h shows the predicted trajectory distribution that satisfies the given goals. The predicted trajectory is sent to the controller of the robot to be executed as detailed next.

Execution and Online Adaptation of the generated movement: The CNMP-driven control loop that aims to bring the robot to the sensorimotor state predicted by the CNMP in each time step is provided in Fig. 3. In each time step, the environment is perceived through sensors of the robot, and the current SM is provided along with the externally defined goals to the CNMP for generation of the sensorimotor values that are expected to be observed to achieve the task. In an online control loop, CNMP predicts the $SM(t)_{t=t_c+1}^{t_n}$ configurations of the next time step and continuously produces motor commands to achieve the predictions. In case the predicted SM and the current SM does not match, i.e. the current SM does not match with the SM that is predicted to match with the corresponding skill, time stops advancing until the error drops below a threshold. With this constantly changing online observation loop, our system can react to the perturbations of the sensor measurements, because of the the low computation complexity of the CNMP at the test time.

Note that the structure of the CNMP does not need any strict requirements on the controller side. In its current form, the motor variables in the generated SM are set as the desired motor commands in each time step, and executed using a PD controller. The fact that the output of the CNMP is constructed as a distribution allows the robot to move freely within the boundaries of the predicted distribution. This flexibility can potentially be exploited to achieve the given task with alternative trajectories. Another option is to use variance information to reason about which segments of trajectory should be followed stiffly and accurately, automatically adjusting the compliance based on the variance in the distribution.

[1]The complexity of the trajectory generation is $\mathcal{O}(n_o + n_q)$ where n_o and n_q are the numbers of the observations and queried time points respectively. Single query takes 0.9 msec. with Inter Core i7 processor and GeForce GTX 1050 gpu, the robot can quickly adapt to the changes in the environment at the test time.

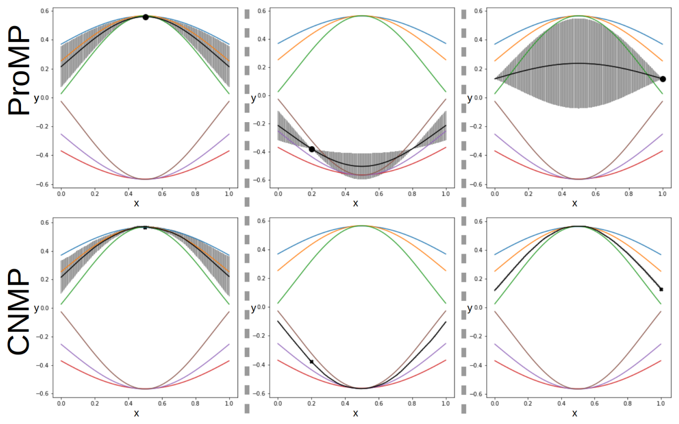


Fig. 4: 2d object avoidance experiment and the results. The demonstrations are given with colored trajectories. The conditions and generated movement distributions are shown with the black dots and gray areas, respectively.

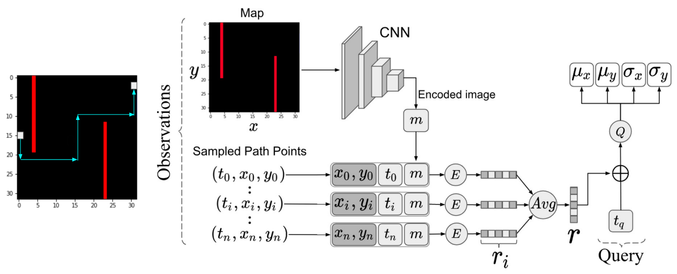


Fig. 5: Left: The 2d obstacle avoidance setup used in learning experiments from top-down images. An example environment and demonstration trajectory were presented. Right: The structure of Convolutional CNMP

IV. Experimental Results

The performance, capabilities, and limitations of the CNMP were studied in two simulated and two real robot experiments. In Section IV.A, the CNMP and another widely used method that encodes distribution of the demonstrations, namely ProMP[15], was compared in learning of a 2d simulated obstacle avoidance task where different modes of the avoidance were demonstrated. In Section IV.B, the capacity of the CNMP to learn in high-dimensional SM was investigated in a 2d obstacle avoidance task where the system learns planning movement trajectory avoiding two obstacles with different using top-down image masks of the environment. In Section IV.C, the performance of the CNMP in learning non-linear environment-trajectory relations in a real robot manipulator was investigated through analyzing generalization capability to environment configurations inside and outside the range of demonstrations. In Section IV.D, the capability of the CNMP model to react to external perturbations detected in the proprioception and force data was verified in the real robot exploiting the learned multi-modal information.

A. *Learning of different modes of operations of the same skill*

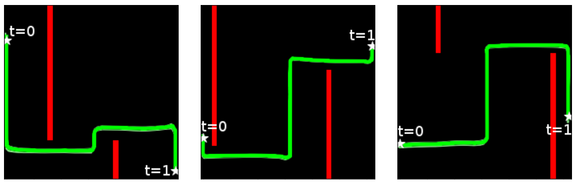
The aim of this experiment is to investigate if and in which conditions the models can automatically discover the categories inherent to the demonstrated primitive; and reproduce new avoidance trajectories taking into account this knowledge. For this purpose, the capabilities of the CNMP and ProMP models were studied in a 2d object avoidance task where demonstrated trajectories passed around either one side or the other side of the obstacle and from which side it passes was not explicitly provided to the system. The task and the demonstrated trajectories are provided in Fig. 4. Each trajectories started from the left side (t=0,(x=0,y=y_0)), it passed around the upper or lower side of the obstacle at time point t=0.5, and ended up on the right side at a symmetric position (t=1,(x=1,y=y_0)). The 6 trajectories shown in the figure with 300 time steps were used to train both models. The ProMP model[2] was trained with 31 basis functions, setting the σ value to 0.05. The CNMP, on the other hand, was trained by sampling a number of observations $\{(t_i, (x_i, y_i))\}_{i=0}^{n\in[1,5]}$ from a randomly selected demonstration in each iteration. Encoder and Query networks were structured in 2 fully connected layers with 128 neurons each. The neural network was trained by using Adam optimizer with an empirically found learning rate of 0.0001.

After training, new avoidance trajectories were generated by setting goals, i.e. by conditioning the models at different points. The obtained results are presented in Fig. 4. The upper and lower rows in this figure correspond to the results obtained by ProMP and CNMP, respectively. In each plot in the figure, the black dot shows the conditioned position. The black line and the shaded area correspond to the mean and variance of the predicted movement distribution computed for the corresponding conditioning. As shown in the first column, when the models were conditioned at a position that ensures avoidance, both models generated distributions and trajectories that enabled the system to avoid the obstacle. On the other hand, when the conditioning was set to other positions such as to an intermediate point (t=0.2,(x=0.2,y=-0.37)) or to an end point (t=1,(x=1,y=0.13)), the ProMP started failing whereas CNMP continued generating movement trajectories that avoid the invisible obstacle. These results show that CNMP learned to produce multiple distributions by observing different categories of the same primitive from the data itself without any explicit parameterization. With this, we conclude that CNMP allows learning and encoding different modes of operation of the same movement primitive, and reliable and flexible reproduction of the trajectories automatically following one of the distinct modes.

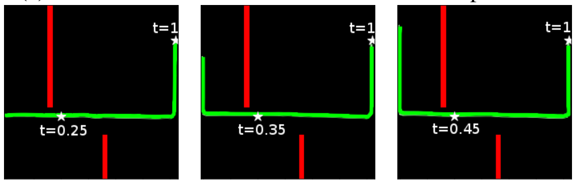
B. *Movement primitive learning in high-dimensional sensorimotor space from top-down raw image masks*

In this section, the capability of CNMPs in learning directly from top-down raw image masks in a simulated experiment

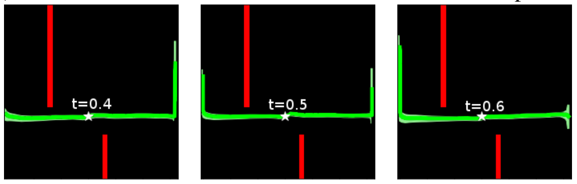
[2]Our implementation and training data for the experiments can be found in https://github.com/rssCNMP/CNMP. To compare our framework with ProMP, ProMp library of the Flowers INRIA Laboratory in learning and generating the movement primitives is used. See https://github.com/flowersteam/promplib



(a) C-CNMP conditioned with start and final positions



(b) C-CNMP conditioned with intermediate and final position



(c) C-CNMP conditioned with one intermediate position

Fig. 6: CNMP was conditioned by (a) start and final positions, (b) intermediate and final positions, and (c) one intermediate position.

setting is studied. Two obstacles of varying sizes were attached to the opposite sides of a 2d room at different positions as shown in Fig. 5. For each room configuration, a motion trajectory that moves the robot from the left to the right side of the room was generated with a simple heuristic. The motion trajectories were presented as demonstrations along with the raw images to train the CNMP model. Convolution operation was included into the model (Fig. 5) to deal with the raw images. Convolution was achieved with a neural network that consists of 4 convolution layers followed by maxpooling operation with channel numbers 8, 16, 32 and 64. At the end, SM vector was composed of 2d position, image mask encoding (m), and the corresponding time-point.

The CNMP model, after trained with 1000 trajectories (in configurations with random start/end positions and randomly placed and sized obstacles) was queried to generate trajectories with different conditions. The conditions were encoded in SM vectors that include time, position and image information. Fig. 6 shows the trajectories generated by the CNMP conditioned with (a) start and end positions, (b) intermediate and end positions, and (c) an intermediate position for a number of test configurations that were not experienced before. As shown, the trained CNMP could successfully generate movement trajectories avoiding the red walls in all configurations. Interestingly, when the CNMP was provided only an intermediate pose at a specified time, it changed the start and end positions, shifting the movement if necessary (Fig. 6c). These results indicate that the CNMP framework was successful in movement primitive learning in high-dimensional sensorimotor space and it is a promising framework for end-to-end learning of complex trajectories.

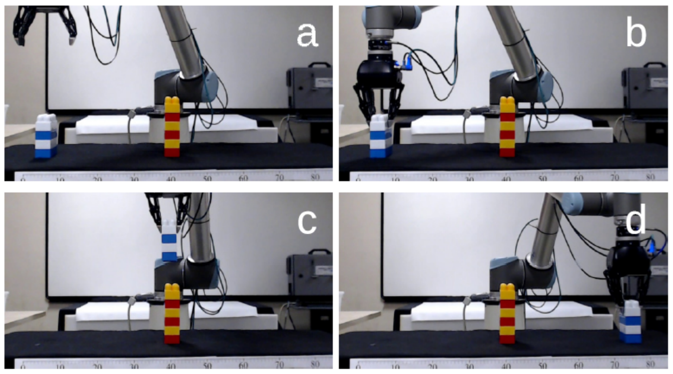


Fig. 7: Experimental setup and snapshots from demonstrations in the obstacle avoidance task where heights were provided as parameters.

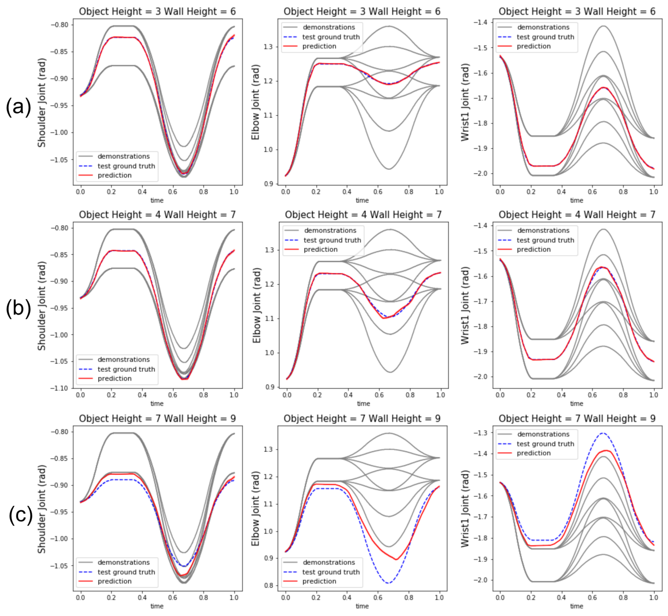


Fig. 8: Qualitative results of selected three joints on novel interpolation (a,b) and extrapolation (c) cases

C. Task Parameterization and Generalization in Real Robot

This experiment investigates the capability of the CNMP in learning of non-linear environment-movement relations in an object pick and place task that requires obstacle avoidance. UR10 robotic arm equipped with a Robotiq gripper and a wrist mounted force/torque sensor was used for this purpose. After the robot was provided a number of demonstrations in different environments, we analyzed the generalization capabilities in novel configurations. The heights of the manipulated object and the static obstacle were provided as task parameters (γ) and the joint angles were included in the SM vector. The task was composed of moving the gripper to a suitable position to pick up the object according to its height first, and then moving it towards the target position following an arc avoiding the obstacle. The experimental setup and snapshots from an

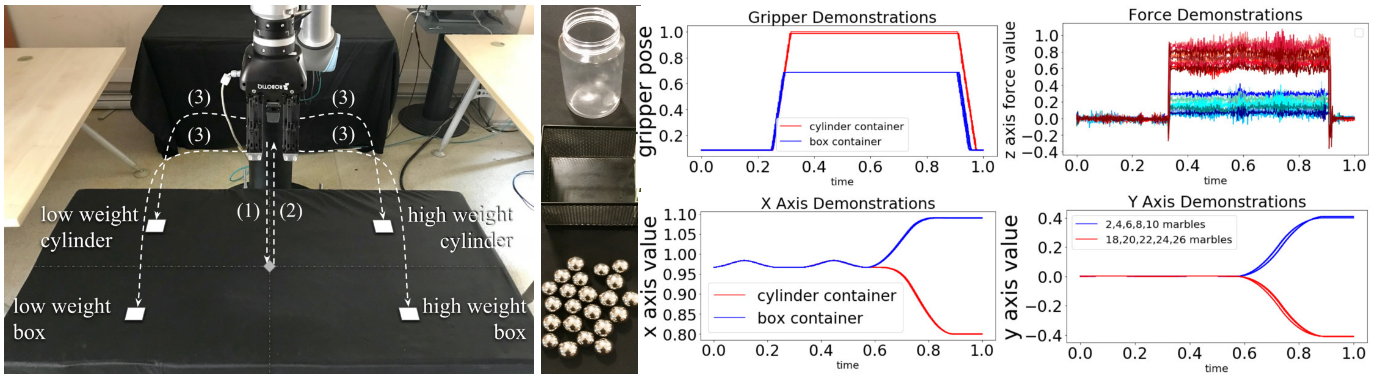


Fig. 9: The experimental setup for pick-and-place task where the placement position depends on the weight and shape of the container. On the left, the movement trajectories for different configurations are provided: The container is reached and picked-up (1), lifted (2), and carried toward target position (3). On the right: demonstration SM data.

example demonstration is provided in Fig. 7. CNMP was trained with 8 movement trajectories that were obtained in different object-obstacle configurations. The heights of the object and obstacle were provided as parameters: γ_o=$(2, 6)$, γ_h=$(2, 4, 6, 8)$. The non-linear relationship between joint values and γ can be observed in Fig 8, which provides the demonstrated trajectories in gray, especially in elbow and shoulder joints.

The generalization capacity of the system to novel object-obstacle settings was tested in two different conditions where the test distributions were sampled from inside and outside of the range of demonstration. Both conditions were tested with three different environments. In the interpolation condition, the robot generated a movement trajectory in the joint space successfully picking and placing the object avoiding the obstacle in all three tasks. The predicted and executed trajectories were compared against ground truth trajectories and the results of two example test cases are provided in the Fig. 8(a,b). The gray lines indicate the demonstrations used for the training, the blue dashed line indicates the ground-truth, and the red line is the prediction of the system. The average absolute errors measured in degrees are also summarized in Table I, where star indicates the novel conditions. The maximum average absolute error along the trajectories was 0.28 degrees. Considering these results, we conclude that the prediction mechanism of the CNMP can generalize to the novel situations sampled inside the training distribution. In the extrapolation cases, our motivation was to gradually increase difficulty and evaluate the limits of the system. For this purpose, the heights of the object and obstacle was gradually increased. The task was successfully executed in $\gamma = (7, 9)$ parameter configuration but the robot started failing when the object and wall height exceeded $\gamma = (8, 10)$. Fig. 8(c) shows the trajectory prediction of one of the trajectories generated and Table I bottom three rows present the average absolute errors for the extrapolation cases. As the results show the performance of the trajectory prediction for extrapolation novel cases decreased significantly, especially when the extremity of the case with respect to the boundaries of the training.

TABLE I: Average joint errors (in degrees) between predicted and ground truth trajectories. The upper and lower three rows show the errors in the interpolation and extrapolation conditions, respectively.

	Joint Errors (deg)					
Task Parameters	**Base**	**Shoulder**	**Elbow**	**Wrist1**	**Wrist2**	**Wrist3**
Obj: 3* Obs.: 3*	0.23	0.271	0.173	0.232	0.064	0.254
Obj: 3* Obs.: 6	0.158	0.167	0.145	0.163	0.061	0.126
Obj: 4* Obs.: 7*	0.099	0.101	0.269	0.280	0.055	0.093
Obj: 7* Obs.: 9*	0.719	0.895	3.569	3.687	0.156	0.665
Obj: 8* Obs.: 10*	1.038	2.325	8.879	8.094	0.360	0.830
Obj: 9* Obs: 11*	1.861	5.380	17.978	14.894	0.590	1.386

D. *Movement Primitive Classification and Adaptation to Perturbations in Real Robot*

This experiment aimed to demonstrate the movement generation capability of the CNMP taking into account the categorical semantics included in the demonstrations and its adaptation capacity in response to external perturbations. The UR10 robot arm, Robotiq gripper and force/torque sensor system was used to teach a pick and place task applied to a container placed on a table. The task is designed with the following schedule: The object was picked up from the middle of the table, and placed at one of the four target locations depending on the type of the container and its weight. Two different containers were used with different amounts of weights as shown in Fig. 9, resulting in four different types (2 containers x 2 weight categories) of demonstrations. 5 different weights were used for each weight category, therefore 4x5=20 demonstrations were provided to the system for training, following the trajectories shown with the dashed lines in the same figure. SM vector included the 3d position of the gripper, the joint positions of the fingers, and the force/torque readings. In testing, given a container (cube with a low weight) unknown to the robot, the robot started the execution without any information on the conditions and updated its predictions online. Before interacting with the container, any four target-points were equally likely for the system, therefore the variance of the prediction started high as shown in the middle-left plot in Fig. 11. The robot

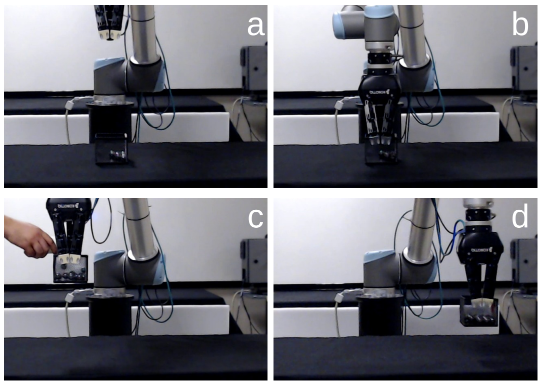


Fig. 10: Movement adaptation experiment. The robot generates the required movement trajectories in its end-effector and finger position spaces, automatically perceiving the task-related features such as the weight and shape of the object through its sensors (proprioception and force/torque readings). https://youtu.be/cPKOIaf0mUc

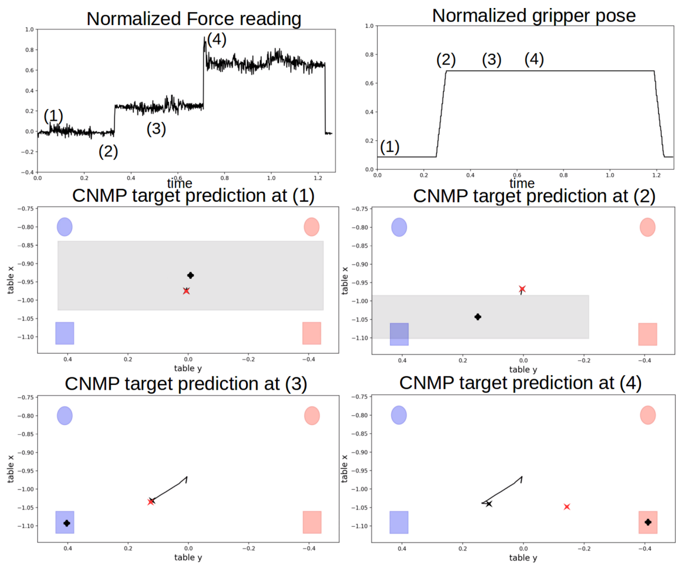


Fig. 11: Online conditioning and adaptation to unexpected changes during motion execution. Final position estimations are shown at 4 timestamps with black dot and gray shade. Container is approached (1), grasped but not lifted (2), lifted and carried (3), made heavier (4).

next reached to the object and started interacting with it: As soon as it enclosed its fingers and started lifting the object, the received feedback from the interacted container through proprioception and force/torque measurements automatically conditioned the robot to generate a trajectory that ends up in the lower-bottom corner from a low-variance distribution (the bottom-left plot in Fig. 11). During the movement of the robot, we stepped in and put more marbles inside the box inside the gripper, effectively increasing the weight of the cube carried by the robot. The updated SM conditioning automatically updated the distribution of the movement trajectory, setting the final point to the bottom-right region on the table (see the bottom-right plot in Fig. 11) and changing the motion of the end-effector towards that position through the PD controller. The increase in the time variable is suspended until the error between predicted and actual SM vanished. Finally, the cube with the increased weight was placed to the position that was demonstrated for high-weight boxes. This experiment showed that although the task parameters (type and weight of the container) were not explicitly provided as parameters, the association between the movement trajectory and these parameters were learned in the SM space of the robot through associating finger proprioception values and force/torque readings with the movement trajectories of different types. Furthermore, thanks to conditioning the system with the current SM in each time step, the system was shown to effectively react to changes that influence the task execution. Such a reaction was possible as the robot inferred that the target position only depended on the sensor measurements and was invariant from the current location of the robot.

V. Conclusion

Our CNMP method was shown to learn the non-linear relationships between environment parameters and complex action trajectories, deal with raw input through convolution operation, learn multiple modes of operations for the same primitive, and generalize to novel configurations if they are sampled from the experienced range. A wide range of relationships were shown to be automatically discovered and learned. In the first real robot pick-and-place task, the robot learned a continuous non-linear relation between the parameters and the action trajectories, whereas in the second robot experiment, it learned that containers with different weights were required to be placed in the same final positions. In the second robot experiment, the factors that influenced the task execution, such as weight and shape of the object, were never provided to the robot. Still, based on its interaction experience, the robot successfully learned to generate the required movement trajectory through exploiting the learned encoding in its low-level sensorimotor space that included proprioception and force readings. Furthermore, our continuous online conditioning with the current sensorimotor values enabled the robot to detect and respond to changes in the factors that influence task execution, on the fly. The PD controller enabled reacting to such external perturbations, however without any convergence or goal-achievement guarantees. In the future, we plan to study on mechanisms that makes search in the execution history and in the encoding space of the sensorimotor representations, and possibly rewinding the taken steps in order to successfully reach to the new environment state exploiting the learned multi-modal relations.

Acknowledgments

This research has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no. 731761, IMAGINE. Also, thanks to Ahmet E. Tekden for early discussions.

References

[1] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Rob. and Auto. Sys.*, 57(5):469–483, 2009.

[2] Tamim Asfour, Pedram Azad, Florian Gyarfas, and Rüdiger Dillmann. Imitation learning of dual-arm manipulation tasks in humanoid robots. *International Journal of Humanoid Robotics*, 5(02):183–202, 2008.

[3] Christopher G Atkeson, Andrew W Moore, and Stefan Schaal. Locally weighted learning for control. In *Lazy learning*, pages 75–113. Springer, 1997.

[4] H Ben Amor, Oliver Kroemer, Ulrich Hillenbrand, Gerhard Neumann, and Jan Peters. Generalization of human grasping for multi-fingered robot hands. In *IROS*, pages 2043–2050. IEEE, 2012.

[5] Sylvain Calinon. A tutorial on task-parameterized movement learning and retrieval. *Intelligent Service Robotics*, 9(1):1–29, 2016.

[6] Sylvain Calinon, Paul Evrard, Elena Gribovskaya, Aude Billard, and Abderrahmane Kheddar. Learning collaborative manipulation tasks by demonstration using a haptic interface. In *Advanced Robotics, 2009. ICAR 2009. International Conference on*, pages 1–6. IEEE, 2009.

[7] Vivian Chu, Ian McMahon, Lorenzo Riano, Craig G McDonald, Qin He, J Martinez Perez-Tejada, Michael Arrigo, Naomi Fitter, John C Nappo, Trevor Darrell, et al. Using robotic exploratory procedures to learn the meaning of haptic adjectives. In *ICRA*, 3048–3055, 2013.

[8] Alain Droniou, Serena Ivaldi, and Olivier Sigaud. Deep unsupervised network for multimodal perception, representation and classification. *Robotics and Autonomous Systems*, 71:83–98, 2015.

[9] Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and S. M. Ali Eslami. Conditional neural processes. In *ICML*, 1704-1713 2018.

[10] Hakan Girgin and Emre Ugur. Associative skill memory models. In *IROS*, pages 6043–6048, 2018.

[11] Alja Kramberger, Andrej Gams, Bojan Nemec, Dimitrios Chrysostomou, Ole Madsen, and Ale Ude. Generalization of orientation trajectories and force-torque profiles for robotic assembly. *Robot. Auton. Syst.*, 98(C), 2017.

[12] Dongheui Lee and Christian Ott. Incremental kinesthetic teaching of motion primitives using the motion refinement tube. *Autonomous Robots*, 31(2-3):115–131, 2011.

[13] Manuel Mühlig, Michael Gienger, and Jochen J Steil. Interactive imitation learning of object movement skills. *Autonomous Robots*, 32(2):97–114, 2012.

[14] R. Pahic, A. Gams, A. Ude, and J. Morimoto. Deep encoder-decoder networks for mapping raw images to dynamic movement primitives. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–6. IEEE, 2018.

[15] Alexandros Paraschos, Christian Daniel, Jan Peters, and Gerhard Neumann. Probabilistic movement primitives. In *NIPS*, pages 2616–2624, 2013.

[16] Peter Pastor, Ludo Righetti, Mrinal Kalakrishnan, and Stefan Schaal. Online movement adaptation based on previous sensor experiences. In *IROS*, 365–371, 2011.

[17] Peter Pastor, Heiko Hoffmann, Tamim Asfour, and Stefan Schaal. Learning and generalization of motor skills by learning from demonstration. In *ICRA*, 763–768, 2009.

[18] Affan Pervez and Dongheui Lee. Learning task parameterized dynamic movement primitives using mixture of gmms. *Intelligent Service Robotics*, 11:61–78, 2018.

[19] Stefan Schaal. Dynamic movement primitives-a framework for motor control in humans and humanoid robotics. In *Adaptive Motion of Animals and Machines*, pages 261–280. Springer, 2006.

[20] Aleš Ude, Andrej Gams, Tamim Asfour, and Jun Morimoto. Task-specific generalization of discrete and periodic dynamic movement primitives. *IEEE Transactions on Robotics*, 26(5):800–815, 2010.

[21] Emre Ugur and Hakan Girgin. Compliant parametric dynamic movement primitives. *Robotica*, 2019. in press.

[22] Sethu Vijayakumar and Stefan Schaal. Locally weighted projection regression: Incr. real time learning in high dimensional space. In *ICML*, pages 1079–1086, 2000.

[23] Y. Zhou and T. Asfour. Task-oriented generalization of dynamic movement primitive. In *IROS*, pages 3202–3209, 2017.