# Learning Robotic Manipulation from Demonstrations by Combining Deep Generative Model and Dynamic Control System

Xiaojian Ma[1,2†], Mingxuan Jing[1†], Fuchun Sun[1*], Huaping Liu[1], Bin Fang[1]

*Abstract*— **Learning and inferring movement from demonstrations is a challenging problem in robotic manipulation due to the simultaneous demand of the endpoint convergence and the trajectory diversity. It could be even more complex when visual sensing is employed, which leads to high dimensionality and large input variance. In this paper, we propose an effective probabilistic method for learning basic movement in robotic manipulation from demonstrations, which can adapt with visual input and variant target tasks at the same while. Our method builds upon a deep generative model, and the trajectory generation problem is formulated as learning and inferring on a directed graphic model which takes the visual features into account, while the endpoint convergence to an expected target position is guaranteed by a second-order dynamic control system. It incorporates the flexibility of the probabilistic model and the stability of the control system, which suits for vision-based long-term planning and robot control. Experimental validations on robotic approaching path planning tasks demonstrate the advantages of our method over the baseline counterparts when only limited demonstrations are available.**

## I. INTRODUCTION

Learning and inferring movement from demonstrations is one of the core ingredients in robotic manipulation. Traditionally, the movement planning and trajectory generation methods are mainly based on purely searching [1] [2] and stochastic optimization [3] [4], they highly depend on accurate hand-crafted models of system dynamics or abstractions of environments, which will be hard to determine from unstructured domestic or industrial scenes in diverse real-world tasks with raw visual sensing. Some recent work models this problem as a decision process and utilizes reinforcement learning from demonstration(RLfD) to learn a policy with the help of demonstrations for specific robotic task [5]–[8], which takes demonstrations as guidance and utilizes the environment information autonomously by exploration. These probabilistic approaches empower the ability to work directly with visual sensing. However, they still have to rely on amounts of explorations that can only be done in simulators due to the safety issues [9]. Most importantly, some essential mechanisms for providing stabilization, goal convergence and safety can be missing as these methods are mostly developed in open-loop control fashion, which introduces challenges for applying them on real-world tasks.

[†]These two authors contributed equally.

[1]Beijing National Research Center for Information Science and Technology (BNRist), State Key Lab on Intelligent Technology and Systems, Department of Computer Science and Technology, Tsinghua University

[2]Center for Vision, Cognition, Learning and Autonomy, Department of Statistics, University of California, Los Angeles (UCLA)

[*]Correspondence to Fuchun Sun (fcsun@tsinghua.edu.cn)

It is wildly expected that successful long-term motion planning can be composed of many desired sub-plannings [10] [11]. Thus we purpose to build a model that can learn and infer basic movements, i.e., approaching, grasping, etc., which should also handle new commands and environment-related information for continuous movement generation. Based on this intuition, in this paper, we propose Contextual Movement Primitives (CMP), a motion planning framework for effective learning and inference of basic movement from demonstrations, while provides the interface for commands(tasks) and environment(context) information as input for long-term planning. By introducing the probabilistic representation of the trajectory, we model the motion planning problem as learning on a directed graphic model. Although the probabilistic distribution of demonstrations and the visual scene can be extremely complicated, with the help of the deep generative model, we make it possible to perform learning and inference on this probabilistic motion planning model. Furthermore, we extend the probabilistic trajectory representation to force space and show an elegant combination with a dynamic control system. The dynamic control system does bring us lots of desired properties on controlling a real robot such as being robust under execution error, guaranteeing the convergence through feedback implied in the dynamic system and producing smooth interpolates, thus significantly improve the quality of generated trajectories.

To summarize, our main contributions are listed below:

- We propose Contextual Movement Primitives(CMP), a probabilistic trajectory generation framework based on the deep generative model, which can handle high-dimension conditional input including visual context and task description, and generate feasible and practical trajectories for robot controlling in reality.
- We combine CMP with a dynamic control system, empower it to directly learn the control value of that system. The control system properties like robust to perturbation and goal-convergence are maintained for dealing with variance targets and endpoints.
- We achieve competitive performance on approaching path planning task both in simulation and real robot than supervised baselines [12] with limited data, which shows that CMP has the potential to be a general framework for robot trajectory and skill learning.

## II. RELATED WORK

Here we will review some previous work about movement primitives and motion planning.

**Probabilistic movement primitives.** In [13], the authors propose a probabilistic representation of trajectory. As it reduces the dimension of the trajectory with arbitrary length into a weight vector of basis function, it parameterizes the trajectory distribution by the distribution of weight and a fixed variance. The authors also show how to satisfy some traditional trajectory constraints including via point, target point and coupling by introducing equality constraints when solving the probabilistic model. Our probabilistic representation of trajectory was basically inherited from this work, however, compared to this work, we extend it to represent the trajectory in force space, which makes it possible to combine the probabilistic model with a dynamic control system. Besides, our CMP framework can learn from the contextual information and task at the same time, while [13] can only learn from the trajectory. Thus our model is more suitable for motion planning on real-world tasks.

**Movement primitive with conditions.** A recent research work [14] introduces a method for learning and retrieving movement primitives with condition input. The main idea of this work is measuring the distance between queries and movement primitives in a pre-built library. Once the metric is learned, we can select a subset of movement primitives with an input query based on the nearest principle, then output the average of the subset. It highly depends on the library of movement primitives, and the inner structure of the trajectory and condition remains unexplored. In contrast, our CMP framework models the conditional distribution of trajectory over contextual information and task. And with the help of the deep generative model, we can directly learn and inference with the complex posterior without any linear assumption [15] and achieve better generalization.

**CNN based movement primitives.** In [12], researchers try to use a forward neural network to learn the parameters of movement primitives directly from condition input(image of the working scene). This model is exactly the same as the NN baseline we used in experiments. As we discussed in Section III-A, a trajectory is inherently to be stochastic instead of deterministic. Thus we believe that a stochastic model may be more appropriate for learning and inference the trajectory, and will lead to better performance. The results of our experiments also support our intuition.

## III. Preliminaries

In this section, several conceptions would be introduced for facilitating further discussions.

### A. Probabilistic Trajectory Representation

Generally, a certain trajectory $\boldsymbol{\tau} = f(\mathbf{t}) : \mathbf{R} \mapsto \mathbf{R^n}$ can be denoted as a time-varying function or sequential-time sampling over that function. For the flexibility on time re-scaling, a monotonous increased phase variable $\mathbf{s} := \mathbf{s}(\mathbf{t}) : \mathbf{R} \mapsto [0, 1]$ could be used instead of actual execution time $\mathbf{t}$, so that the trajectory could be rewritten as $\boldsymbol{\tau} = f(\mathbf{s})$.

As described above, $f(\mathbf{s})$ could be any continuous function, which is catastrophic for learning directly. For a smooth trajectory whose values at each time point are closely co-related with the value in nearby time phase, a reasonable method may be approximating $f(\mathbf{s})$ by limited and fixed number of parameters using Radial Basis Function(RBF) approximater. RBF uses kernel functions $\boldsymbol{\Phi}^T(\mathbf{s}) = [\boldsymbol{\phi}_1(\mathbf{s}), \boldsymbol{\phi}_2(\mathbf{s}), \cdots]$ as basic components and regard mixture weights $\mathbf{w}$ as parameters. $\boldsymbol{\tau} = f(\mathbf{s}) = \boldsymbol{\Phi}^T(\mathbf{s})\mathbf{w} + \boldsymbol{\epsilon}_\tau$, where $\boldsymbol{\epsilon}_\tau \sim N(0, \boldsymbol{\Sigma}_\tau)$ denotes approximation residuals.

When modeling the property of trajectory variability under perturbations [16] on certain environment, a trajectory could be seen as composed by deterministic kernel functions at each time phase but with stochastic mixture weight $\mathbf{w}$ drawn from a distribution. For most common case, Gaussian distribution is used here: $\mathbf{w} \sim N(\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w)$, and the probability of a trajectory $\boldsymbol{\tau}$ could be calculated as:

$$\begin{aligned} p(\boldsymbol{\tau}) &= \prod_s p(f(\mathbf{s})|\mathbf{w})p(\mathbf{w}) \\ &= \prod_s N\left(f(\mathbf{s})|\boldsymbol{\Phi}^T(\mathbf{s})\mathbf{w}, \boldsymbol{\Sigma}_\tau\right) N\left(\mathbf{w}|\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w\right) \end{aligned} \tag{1}$$

As described in [13], here $\boldsymbol{\Sigma}_\tau$ is a fixed value once weight $\mathbf{w}_\tau$ is calculated with given $\boldsymbol{\Phi}(\mathbf{s})$ on $\boldsymbol{\tau}$. The optimal $\mathbf{w}_\tau$ can be calculated by maximum logistic likelihood estimation:

$$\begin{aligned} &\max_w \log p(\boldsymbol{\tau}|\mathbf{w}) = \\ &\max_{w,\Sigma_\tau} \sum_s \log \frac{\exp\left(-\frac{1}{2}(\boldsymbol{\tau} - \boldsymbol{\Phi}^T(\mathbf{s})\mathbf{w})^T \boldsymbol{\Sigma}_\tau^{-1}(\boldsymbol{\tau} - \boldsymbol{\Phi}^T(\mathbf{s})\mathbf{w})\right)}{|\boldsymbol{\Sigma}_\tau|\sqrt{(2\pi)^{dim(\boldsymbol{\tau})}}} \end{aligned} \tag{2}$$

$$\Rightarrow \max_w \log p(\boldsymbol{\tau}|\mathbf{w}) = \min \sum_s \left\| \boldsymbol{\tau} - \boldsymbol{\Phi}^T(\mathbf{s})\mathbf{w} \right\|_2 \tag{3}$$

Since the trajectory variance $\boldsymbol{\Sigma}_w$ is fixed, and $N(\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w)$ is isotropic, (2) could then be rewritten as (3). Now a trajectory $\boldsymbol{\tau}$ can be regarded as a sample draw from distribution shown in (1) with parameter $\boldsymbol{\mu}_w$ and $\boldsymbol{\Sigma}_w$, which can be learned and inferred under probabilistic model. Given $\mathbf{w} \sim N(\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w)$, a trajectory can be generated by computing $E_{\boldsymbol{\tau} \sim p(\boldsymbol{\tau}|\mathbf{w})}[\boldsymbol{\tau}] = \boldsymbol{\Phi^T}(\boldsymbol{s})\mathbf{w}$. In the rest of this paper, we will consider $\mathbf{w}$ and its distribution $p(\mathbf{w})$ as the **trajectory probabilistic representation**.

For trajectory learning, usually, we are more interested in whether a trajectory can be generated with some conditions $\mathbf{c}$ that related to a specific task. This is exactly what we concentrate on in this paper. Thus we will discuss the details about learning $p(\mathbf{w}|\mathbf{c})$ instead of $p(\mathbf{w})$ in the following.

### B. Dynamic Control System

A trajectory generator or movement controller should have the ability to handle the dynamic properties and perturbations when applying on real-world robots. Overall, a control system should hold these characteristics: 1) keeping the trajectory dynamically feasible, so that the trajectory can be executed on real-world; 2) guaranteeing to reach target point at the end of the trajectory, so that the task could be accomplished; 3) being robust on noise or robot control errors so that the performance could be reliable; 4) the control signal can be efficiently learned and generated.

A well designed second-order dynamic control system with force signal $f_{ctl}(\mathbf{s}) : \lim_{\mathbf{s}\to 1} f_{ctl}(\mathbf{s}) = 0$ and end constraint $\mathbf{g}$, would properly satisfy these requirements and generate trajectories by solving the system dynamic:

$$\ddot{\boldsymbol{\tau}} = \mathbf{K}(\mathbf{g} - \boldsymbol{\tau}) - \mathbf{B}\dot{\boldsymbol{\tau}} + f_{ctl}(\mathbf{s}; \mathbf{w}) \tag{4}$$

$\mathbf{K}$, $\mathbf{B}$ are constant system parameters that control the time domain attributes of this dynamic system. This kind of dynamic system is analytically well understood that holds stability properties, which means when integrated, it would generates the desired behavior under a given $f_{ctl}$ and $\mathbf{g}$.

An interesting fact is, here $f_{ctl}(\mathbf{s})$ could be regarded as a force space trajectory and thus we can describe it with the same approach as a normal trajectory(mentioned in Section III-A). This means that it is possible to alternatively represent and learn the force space trajectory instead of the common trajectory with position, then control the robot through a dynamic control system. As we discussed above, by learning force space trajectories, we can obtain some control-friendly properties brought by the dynamic control system [17]. More relevant details will be discussed in the following.

## IV. Methogology

### A. Problem Formulation

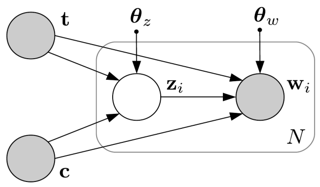

Fig. 1. Graphical model underlying our approach. Shaded nodes are all observed during learning phase, and our model learns parameters $\boldsymbol{\theta} = \{\boldsymbol{\theta}_z, \boldsymbol{\theta}_w\}$. During testing, only $\mathbf{c}$, $\mathbf{t}$ are available.

We denotes an individual trajectory representation as $\mathbf{w}$, contextual information $\mathbf{c}$, and a descriptor to the given task as $\mathbf{t}$. Since there can be multiple feasible trajectories that comply with the same given task and planning context, i.e., target positions, obstacles and auxiliary constraints. We thus introduce a set of continuous latent variables $\mathbf{z}$ with prior distribution $p_{\boldsymbol{\theta}_z}(\mathbf{z}|\mathbf{c}, \mathbf{t})$ to capture this inherent trajectory diversity, and apparently, they should depend on some factors related to the planning context for execution feasibility. Finally, we can define the posterior conditional distribution of $w$ as $p_{\boldsymbol{\theta}_w}(\mathbf{w}|\mathbf{c}, \mathbf{t}, \mathbf{z})$. A visual illustration of the underlying graphical model is shown in Figure 1. Given contextual information and task taken from demonstrations $D = \{(\mathbf{w}_1, \mathbf{c}_1, \mathbf{t}_1), (\mathbf{w}_2, \mathbf{c}_2, \mathbf{t}_2)...(\mathbf{w}_n, \mathbf{c}_n, \mathbf{t}_n)\}$, the problem will be learning on this directed probabilistic model with maximum likelihood, in presence of continuous latent variables of $p_{\boldsymbol{\theta}_z}$ with intractable posterior distribution $p_{\boldsymbol{\theta}_w}$. $\boldsymbol{\theta} = \{\boldsymbol{\theta}_z, \boldsymbol{\theta}_w\}$ will be the parameters of the expected generative model. When the model is learned, a feasible trajectory under specified context and task can be inferred on it.

**Representation of contextual information and task.** In our framework, $\mathbf{c}$ can be varied. But here we simply provide the RGB photograph taken from the First-Person view of the working scene, where the trajectory will be executed, as contextual information. As for the task descriptor, we assume that there exists a set $\mathbb{T}$ with finite tasks. Then $\mathbf{t}$ is a one-hot tensor of size $|\mathbb{T}|$. Each bit in $\mathbf{t}$ is associated with a single task. Thus we can see $\mathbf{t}$ as a random variable derived from $\mathbf{t} \sim \text{Categorical}(n = 1, p_0 = ... = p_k = 1/k)$.

### B. Model Overview

To learn and infer a better movement for robotic trajectory-based planning, we propose Contextual Movement Primitives(CMP), a probabilistic trajectory representation and learning framework that effectively generates trajectories for specific context and task in an end-to-end manner. By combining CMP with a dynamic control system, a robot can operate under the control value output by our framework directly, and acquire some properties of dynamic control system like perturbation-robust and goal-convergence, which are extremely important for robots and tasks in real-world.

Our CMP framework consists of 3 modules: CMP generator(CMPG), localization network(LN) and motion control system(MCS). An overview of it is shown in Figure 2. All the modules composite CMP together to enable end-to-end movement learning and inference. We will show the details of each module below, and provide the full specification in the Appendix [1].

### C. Movement Inference and Trajectory Generation

In our framework, CMP generator(CMPG) serves as a generative model that receives the contextual information $\mathbf{c}$ and task descriptor $\mathbf{t}$ as input, then performs an inference on the learned graphical model about $p_{\boldsymbol{\theta}_w}(\mathbf{w}|\mathbf{c}, \mathbf{t}, \mathbf{z})$ to produce a representation $\mathbf{w}$ of trajectory. However, learning on this graphical model can be challenging, since latent variable $\mathbf{z}$ is continuous and $p_{\boldsymbol{\theta}}$ is intractable. We solve this issue following Welling *et.al.* [18] and Lee *et.al.* [19] by maximizing the *evidence lower bound*(ELBO) $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}, \mathbf{w}_i, \mathbf{c}_i, \mathbf{t}_i)$ of the likelihood over demonstrations $D$(Section IV-A):

$$\begin{aligned} &\log p_{\boldsymbol{\theta}}(\mathbf{w}_i|\mathbf{c}_i, \mathbf{t}_i) \geq \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}, \mathbf{w}_i, \mathbf{c}_i, \mathbf{t}_i) \\ =&\mathbb{E}_q \left[ -\log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{w}_i, \mathbf{c}_i, \mathbf{t}_i) + \log p_{\boldsymbol{\theta}_w}(\mathbf{w}_i|\mathbf{z}, \mathbf{c}_i, \mathbf{t}_i) \right] \qquad (5) \\ =&- KL\left( q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{w}_i, \mathbf{c}_i, \mathbf{t}_i) || p_{\boldsymbol{\theta}_z}(\mathbf{z}|\mathbf{c}_i, \mathbf{t}_i) \right) \\ &+ \mathbb{E}_q \left[ \log p_{\boldsymbol{\theta}_w}(\mathbf{w}_i|\mathbf{z}, \mathbf{c}_i, \mathbf{t}_i) \right] \qquad (6) \end{aligned}$$

Notice that the lower bound above is slightly different from [18]. It is extended to a conditional distribution. We provides an mechanism closed to [19] under encoder-decoder structure to differentiate and optimize the conditional lower bound $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}, \mathbf{w}_i, \mathbf{c}_i, \mathbf{t}_i)$ w.r.t. both the variational parameters $\boldsymbol{\phi}$ and generative parameters $\boldsymbol{\theta}$. The model is illustrated in Figure 2 with an encoder, a decoder and a prior network. Encoder is a neural network that maps $\mathbf{c}$, $\mathbf{t}$ and $\mathbf{w}$ into latent space $\mathbf{z}_i \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{w}_i, \mathbf{c}_i, \mathbf{t}_i)$ with parameters $(\boldsymbol{\mu}_e(\mathbf{w}_i, \mathbf{c}_i, \mathbf{t}_i), \boldsymbol{\Sigma}_e(\mathbf{w}_i, \mathbf{c}_i, \mathbf{t}_i))$. While decoder take the input of $\mathbf{z}_i$ from encoder to approximate the posterior $p_{\boldsymbol{\theta}_w}$. The first term in (5) measure how close the distribution $q_{\boldsymbol{\phi}}$ to $p_{\boldsymbol{\theta}_z}$,

[1]Supplementary materials containing full results can be found at https://github.com/conference-supplementary/ICRA20-CMP-Supp.

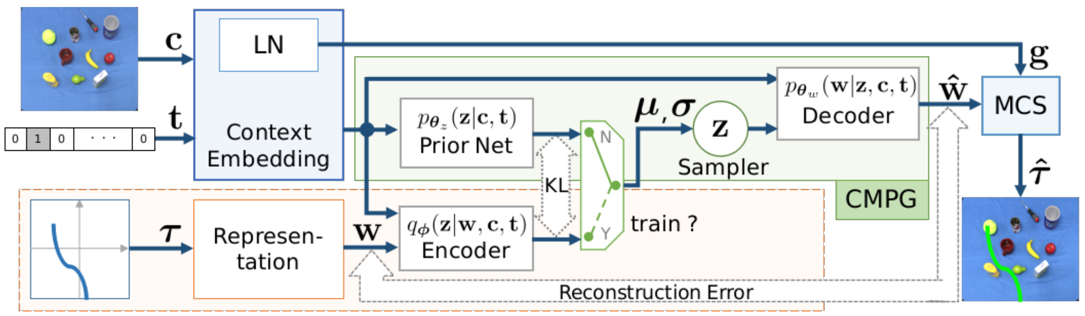
c
t
LN
Context Embedding
$p_{\theta_z}(\mathbf{z}|\mathbf{c},\mathbf{t})$ Prior Net
$q_\phi(\mathbf{z}|\mathbf{w},\mathbf{c},\mathbf{t})$ Encoder
$p_{\theta_w}(\mathbf{w}|\mathbf{z},\mathbf{c},\mathbf{t})$ Decoder
$\boldsymbol{\mu},\boldsymbol{\sigma}$
z
Sampler
KL
N
Y
train ?
CMPG
g
$\hat{\mathbf{w}}$
MCS
$\hat{\boldsymbol{\tau}}$
$\boldsymbol{\tau}$
Represen-tation
w
Reconstruction Error

Fig. 2. Illustration of our model. Besides CMPG, LN and MCS, an encoder is needed during training. The inner structure of Context Embedding and LN can be found in Figure 3.

while the later one will be represented by a neural network named prior network. Thus the KL term can be computed in closed-form. The second term of expectation over $q_\phi$ can be formalized by concatenating the encoder and decoder, then minimizing the reconstruction error.

During training, reparameterization trick [18] will be used to re-write the expectation w.r.t. $q_\phi$ in (5). In each iteration, the encoder will generate the parameter $(\boldsymbol{\mu}_e, \boldsymbol{\Sigma}_e)$ of $q_\phi$, then a reparameterization of latent variable $\mathbf{z} = \boldsymbol{\mu}_e + \boldsymbol{\Sigma}_e\boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim N(\mathbf{0}, \mathbf{I})$ will be fed into decoder. Such that the Monte Carlo estimation of the second term in (5) is differentiable w.r.t. $\phi$. When all the parameters are learned, the decoder will be reserved as CMPG, it will generate a trajectory representation $\mathbf{w}$ with contextual information $\mathbf{c}$, task descriptor $\mathbf{t}$ and latent $\mathbf{z} \sim p_{\theta_z}(\mathbf{z}|\mathbf{c}_i, \mathbf{t}_i)$. This representation $\mathbf{w}$ will be fed into MCS later to recover the real trajectory or control value for operating the robot.

Since encoder, decoder and prior network will all take $\mathbf{c}$, and $\mathbf{t}$ as input, a two-stream structure [20] that designed for handling these data is needed. As illustrated in Figure 2, this two-stream structure is placed before all three modules. One stream is a convolutional neural network(CNN) that maps the RGB image $\mathbf{c}$ of working scene to an embedding that will be used for predicting the distribution over the latent variable $\mathbf{z}$. For another stream handling task descriptor $\mathbf{t}$, we will concatenate it with the embedding of $\mathbf{c}$. All the parameters between these two-stream structures are shared.

**Attention with Spatial Softmax.** Prior work on learning visual-motor policy [21] showed that, Spatial Softmax on the feature map of the last CNN layer will be helpful for learning useful representation about object position, which can be particularly important in robotics domains [22] [21] [23]. Spatial Softmax is a operation that converts the channel-wise feature $\mathbf{a}_{cij}$ into probabilistic distribution $\mathbf{s}_{cij}$, then extract the expected position $(\mathbf{f}_{cx}, \mathbf{f}_{cy})$ in coordinate representation:

$$\mathbf{s}_{cij} = \exp \mathbf{a}_{cij} / \sum_{i'j'} \exp \mathbf{a}_{ci'j'} \tag{7}$$

$$\mathbf{f}_{cx} = \sum_{ij} \mathbf{s}_{cij}\mathbf{x}_{ij} \quad ; \quad \mathbf{f}_{cy} = \sum_{ij} \mathbf{s}_{cij}\mathbf{y}_{ij} \tag{8}$$

We consider it as an attention mechanism for trajectory learning. Following prior work [21], and in another implementation of CMPG, we use these feature points $(\mathbf{f}_{cx}, \mathbf{f}_{cy})$ extracted by Spatial Softmax instead of the origin feature as the embedding of $\mathbf{c}$. The experiments show that this extra operation can significantly improve the qualities of generated trajectories.

### D. Target Localization for Position Constraints

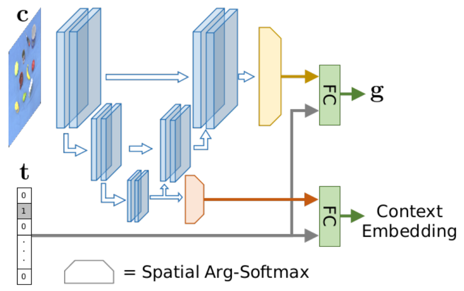

Fig. 3. The inner structure of Context Embedding and LN. Some layers are shared between these 2 models.

Most of the tasks in robotics domains are target-centric [24] [25] [26], especially for trajectory learning tasks. This means that accurate position information of task-related objects can be extremely important. Besides the implicit mechanism(Spatial Softmax) in CMPG for learning position-aware representation, we also introduce the localization network(LN) to explicitly predict the position information $d$ of object-of-interest in the current task. Such information will be used as the goal constraints $\mathbf{g}$ in MCS.

In general, LN will take contextual information $\mathbf{c}$(need to be an image) with task descriptor $\mathbf{t}$ as input, and predict an objectness map. Notice that here we suppose there will be only one object-of-interest for each trajectory generation procedure; thus the regressed position $\mathbf{g} = (\mathbf{f}_{objx}, \mathbf{f}_{objy})$ could then be obtained by performing a Spatial Softmax over the objectness map. In our implementation, LN reuses some convolutional layers with the two-stream structure in CMPG for handling contextual information. Figure 3 illustrates the details of LN and feature extractor in CMPG. The training of LN is quite similar to other saliency detection models [27] [28]. However, here we only provide a bounding box of object-of-interest as the ground truth saliency mask.

### E. Trajectory Execution Under Dynamic Control System

Supposed that the representation of trajectory $\mathbf{w}$ and target object position(or namely, goal) $\mathbf{g}$ are ready. However, we still cannot directly execute them on a robot. MCS is proposed to control the real system with these pieces of information. In our framework, MCS can operate under two modes. The main differences between them include how a dynamic control system mentioned in Section III-B integrates into the trajectory execution and the type of trajectory that is represented by $\mathbf{w}$.

**MCS as a trajectory solver.** For some relatively easy trajectory learning task, a straight-forward approach is directly representing the trajectory of position. In this case, MCS is simply identical as performing a trajectory recovery with $\boldsymbol{\tau} = \mathbb{E}_{p(\boldsymbol{\tau}|\mathbf{w})}[\tau] = \boldsymbol{\Phi}^T(\mathbf{s})\mathbf{w}, \mathbf{w} \sim N(\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w)$ as shown in Section III-A. Once the $\mathbf{w}$ is generated, the corresponding trajectory $\boldsymbol{\tau}$ can be computed; then any position control methods can be applied to control a robot to follow it.

**MCS as a dynamic control system.** Although a direct representation of a trajectory of position can be convenient, this approach will suffer from some problems in real-world robot controlling. For example, the shape and endpoint of a generated trajectory are treated equally in direct representation, but for manipulation tasks, the convergence of generated trajectory to a task-specified goal should be guaranteed. Also, when the robot failed to follow the trajectory or a burst perturbation occurs, a control policy should be designed for following the original executing path while keeping the trajectory smooth. We solve these issues by combining CMP with a dynamic control system mentioned in Section III-B, which will be integrated into CMP as MCS. To control a robot with this version of MCS, we first need to compute the probabilistic representation of force space trajectory $\mathbf{w}$ from a demonstrated trajectory of position $\boldsymbol{\tau}$, which is equal to solving the optimization problem below:

$$\min_{w} \left\| \boldsymbol{\Phi}^T(\mathbf{s})\mathbf{w} - \ddot{\boldsymbol{\tau}} - \mathbf{K}(\mathbf{g} - \boldsymbol{\tau}) + \mathbf{B}\dot{\boldsymbol{\tau}} \right\|_2 \quad (9)$$

where $\mathbf{g}$ is the goal of the current trajectory(for simplification, we use the last point of $\boldsymbol{\tau}$). Training this version of CMPG(with force space trajectory) is basically the same as CMPG with a trajectory of position. After the model is converged, force trajectory representation $\mathbf{w}$ can be generated by CMPG after $\mathbf{c}$ and $\mathbf{t}$ are provided. The goal constraints $\mathbf{g}$ can also be obtained from LN simultaneously.

Now we can control our robot with system dynamics described in (4). For each phase step $\mathbf{s}_i$, we can compute the desired position of a robot in the next step by performing the numerical integration over the system dynamics (4) with generated force trajectory representation $\mathbf{w}$. This operation can be performed iteratively until we accomplish all the movement and reach the goal of $\mathbf{g}$. Even when a single movement is failed to execute correctly, i.e., perturbed by others, the execution in the next step will not be affected, and the goal constraint can still be guaranteed.

## V. EXPERIMENT

We conduct experiments with the Approaching Path Planning task, which will be described in the following. These experiments are designed for answering the questions below:

- Can our method effectively represent the trajectory and learn the posterior with contextual information and task?
- How does the explicit approach(LN) compare to the implicit mechanism(Spatial Softmax) on position extraction for learning trajectory under target-centric task?
- How does combining dynamic control system(MCS as a dynamic control system) compare to recovering trajectory directly(MCS as a trajectory solver)?

For the first question, we prepare environments that cover different domains, including a simulated environment base on Gazebo [29] simulator and an environment in real-world. We will compare the performance of our CMP framework against other baselines on learning and generalization with limited trajectory demonstrations in provided environments. As for the demonstrations $D$, we first synthesis the working scene by randomly selecting and placing the objects, then the trajectories are generated by traditional planners [1] [2] or simply specified by a human. To make the trajectories more similar to biological movement, zero-mean Gaussian noise with cosine value over the phase will be added. For the latter two questions, we will do ablation analysis by temporarily removing some of the modules in our CMP framework, then test the modified models on a real robot.

### A. Approaching Path Planning Task

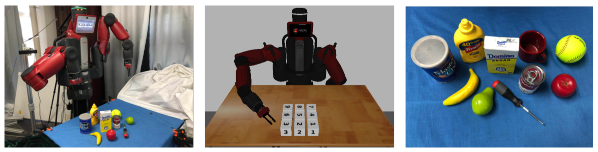
Fig. 4. Experiment setup. Left: In real-world. Middle: In simulation. Right: Selected object set.

We first introduce a set of tasks that will be used in our experiments: **Approaching Path Planning**. For each task, a 7-DOF Baxter manipulator is controlled to approach a specified target among a cluster of novel objects, pick it up, then back to the initial position. For simulation, we use nine cubes with number 1-9 on them. For the real-world task, all the objects are selected from a subset of YCB object and model set [30]. Experiment setup for simulation/real world and the subset of YCB are shown in Figure 4. The generalization to an unseen task is non-trivial due to the arbitrary amount and position. This requires the learned model to adapt to totally different contextual information (a different working scene with different objects) and tasks (different specified target). Furthermore, compared to decision-process based methods [31], our model needs to generate a complete trajectory, which is even harder due to the length and the smooth requirements for feasibility.

### B. Trajectory Representation, Learning, and Inference

In this experiment, we trained the proposed model(CMP) and the baseline deep neural networks on a synthesized demonstration set. Then test them on another demonstration set with the same size as the training set. We compare the estimations of conditional likelihoods(CLLs) on the testing set of 2 different environments(Simulation and YCB). The early stopping is used during the training based on the estimation of CLLs on the validation set.

We visualize the generated trajectories in Figure 5 as qualitative analysis. Comparing to the single deterministic

TABLE I

THE CLLS ON SYNTHESIZED DATASET WITH DIFFERENT SIZES.

| CLL | $\|D\|$ =1000 | | $\|D\|$ =2000 | | $\|D\|$ =5000 | |
|---|---|---|---|---|---|---|
| | Simulation | YCB | Simulation | YCB | Simulation | YCB |
| NN(baseline) | 1.582 | 4.487 | 2.074 | 4.468 | 2.242 | **5.634** |
| CMP($\mathbf{z} \sim N(\mathbf{0}, \mathbf{I})$) | 1.674 | 4.460 | 2.093 | **4.634** | 3.866 | 5.527 |
| CMP($\mathbf{z} \sim p_{\boldsymbol{\theta}}(\mathbf{c}, \mathbf{t})$) | **1.704** | **4.490** | **2.149** | 4.581 | **5.434** | **5.634** |

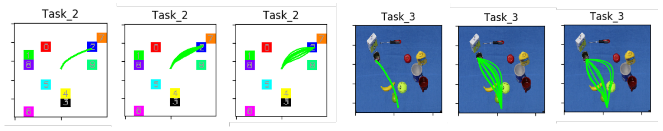

Fig. 5. Visualization of generated trajectories and working scene(in synthesized simulation/YCB dataset). For each dataset, left is NN, middel is CMP with Gaussian prior and right is normal CMP.

prediction made by NN, CMP can generate more diverse and robust trajectories for a specified task. Quantitative results are shown in Table I. We observe that the estimated CLLs of CMP framework outperforms the baseline NN. Here we also compare an alternative of CMP, which samples latent variables simply from an isotropic multivariate Gaussian prior instead of the true latent distribution $p_{\boldsymbol{\theta}}$ during both training and inference. According to the results, the origin CMP achieves better performance than the alternative in most of the experiments, which shows that the latent variables are indeed conditional correlated with context and task.

### C. Ablation Analysis on Real Robot

**Experiment setup.** We use a 7-DOF Baxter robot as the main platform. Ten objects are used from the YCB object set. The size of the working area is 90cm×90cm with a blue background. A Logitech C170 camera is placed over the top of the working scene to capture the context information. The entire system runs on ROS [32] and the concrete motion in joint space is computed via inverse kinematics. For each experiments group, we train the model on a synthesized dataset with size 5000, then evaluate it on Baxter on 50 different task configurations. Each object was tested approximately 30 times.

**Evaluation criteria.** While there might be multiple ways to approach a specified object, not all the trajectories are feasible for execution. Since we use inverse kinematics to compute the motion in joint space, some trajectories may not have valid corresponding movement(which we called *unfeasible*). As a result, it is natural to separate the robot executions into three different categories:

1) Success, where the robot successfully executes a motion and finish the specified task.
2) Failed due to the unfeasible trajectory.
3) Failed due to the wrong trajectory.

**Performance evaluation.** We set up three groups of experiments. In the first group, both Spatial Softmax(in CMPG) and LN is removed. Without target constraints provided by LN, MCS will work as a trajectory solver. In the second group, only LN is removed, while MCS will remain as a trajectory solver. For the last group, we will use a complete CMP with a dynamic control system. All the other configurations will be the same among these groups besides the model.

Table II shows the results of our ablations and an NN baseline. There are several interesting facts in these results. First, we can see that the Spatial Softmax operation in CMPG is extremely important since it yields a significant improvement. Secondly, CMP − LN is exactly the third model in Table I. Compared to the NN baseline, the latter one is more likely to generate unfeasible trajectories, which shows that the proposed probabilistic model is more appropriate for trajectory learning. we can also find that the results between objects are not remarkably different. This means our model is successfully trained to focus more on object position instead of other irrelevant informationAt last, the combination of a dynamic control system can also yield improvements on the results as it provides the guarantee of reaching the goal. However, in our experiments, we find that the smoothness of weight $\mathbf{w}$ when learning the force space trajectory is not as good as position trajectory, which makes the model hard to converge when training CMP, and causes some unfeasible or even wrong trajectories. How to balance this trade-off will be worth some further research.

TABLE II

RESULTS OF BASELINES AND ABLATIONS OF PROPOSED METHOD.

| Evaluation | Success | Failure (unfeasible) | Failure (wrong) |
|---|---|---|---|
| NN(baseline) | 39.7% | **14.0%** | 46.3% |
| CMP − SS* − LN | 24.3% | 12.3% | **63.4%** |
| CMP − LN | 56.7% | 2.3% | 41.0% |
| CMP | **75.7%** | 13.0% | 11.3% |

*denotes Spatial Softmax

## VI. CONCLUSION

In this work, we presented CMP, a novel framework aiming at a challenging problem in robotic manipulation: learning feasible movement on a specified task and corresponding contextual information from demonstrations. By elegantly combining the generative model and dynamic control system, our method makes it possible to work with raw visual input while guaranteeing the stabilization and convergence of movement. Experiments on both simulation and real-world shows that our approach can learn and inference movement effectively and stable. There are a lot of exciting directions for future work. One of them could be combining CMP with other sequential models to make long-term movement inference. Most importantly, we hope to apply our method to more challenging robotic tasks and explore its potential towards a general robotics trajectory and skill learning system.

## ACKNOWLEDGMENT

This research was funded jointly by the National Science Foundation of China (NSFC) and the German Research Foundation (DFG) in project Cross Modal Learning, NSFC 61621136008/DFG TRR-169. It was also partially supported by National Science Foundation of China (Grant No.91848206, U1613212).

## References

[1] S. Karaman and E. Frazzoli, “Sampling-based algorithms for optimal motion planning,” *The international journal of robotics research (IJRR)*, 2011.

[2] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, “Probabilistic roadmaps for path planning in high-dimensional configuration spaces,” *IEEE transactions on Robotics and Automation*, 1996.

[3] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, “Stomp: Stochastic trajectory optimization for motion planning,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[4] J. Kennedy and R. Eberhart, “Particle swarm optimization,” in *IEEE International Conference on Neural Networks*, 1995.

[5] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, I. Osband, *et al.*, “Deep q-learning from demonstrations,” in *AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

[6] T. Brys, A. Harutyunyan, H. B. Suay, S. Chernova, M. E. Taylor, and A. Nowé, “Reinforcement learning from demonstration through shaping,” in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2015.

[7] J. Chemali and A. Lazaric, “Direct policy iteration with demonstrations.” in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2015.

[8] T. Cederborg, I. Grover, C. L. Isbell, and A. L. Thomaz, “Policy shaping with human teachers.” in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2015.

[9] B. Eysenbach, S. Gu, J. Ibarz, and S. Levine, “Leave no trace: Learning to reset for safe and autonomous reinforcement learning,” in *International Conference on Learning Representations (ICLR)*, 2018.

[10] L. P. Kaelbling and T. Lozano-Perez, “Integrated task and motion planning in belief space,” *International Journal of Robotics Research (IJRR)*, 2013.

[11] R. Platt, R. Tedrake, L. Kaelbling, and T. Lozano-Perez, “Belief space planning assuming maximum likelihood observations,” in *Robotics: Science and Systems (RSS)*, 2010.

[12] A. Pervez, Y. Mao, and D. Lee, “Learning deep movement primitives using convolutional neural networks,” in *IEEE-RAS International Conference on Humanoid Robots (Humanoid)*, 2017.

[13] A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann, “Probabilistic movement primitives,” in *Advances in neural information processing systems (NIPS)*, 2013.

[14] Y. Zhou and T. Asfour, “Task-oriented generalization of dynamic movement primitive,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[15] A. Paraschos, C. Daniel, J. Peters, and G. Neumann, “Using probabilistic movement primitives in robotics,” *Autonomous Robots*, 2018.

[16] E. A. Rückert, G. Neumann, M. Toussaint, and W. Maass, “Learned graphical models for probabilistic planning provide a new class of movement primitives,” *Frontiers in computational neuroscience*, vol. 6, p. 97, 2013.

[17] P. Pastor, M. Kalakrishnan, S. Chitta, E. Theodorou, and S. Schaal, “Skill learning and task outcome prediction for manipulation,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[18] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” in *International Conference on Learning Representations (ICLR)*, 2014.

[19] K. Sohn, H. Lee, and X. Yan, “Learning structured output representation using deep conditional generative models,” in *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in neural information processing systems (NIPS)*, 2012.

[21] S. Levine, C. Finn, T. Darrell, and P. Abbeel, “End-to-end training of deep visuomotor policies,” *The Journal of Machine Learning Research (JMLR)*, 2016.

[22] C. Finn, X. Y. Tan, Y. Duan, T. Darrell, S. Levine, and P. Abbeel, “Deep spatial autoencoders for visuomotor learning,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[23] T. Yu, C. Finn, A. Xie, S. Dasari, T. Zhang, P. Abbeel, and S. Levine, “One-shot imitation from observing humans via domain-adaptive meta-learning,” *Robotics: Science and Systems (RSS)*, 2018.

[24] C. Finn and S. Levine, “Deep visual foresight for planning robot motion,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[25] D. Pathak*, P. Mahmoudieh*, M. Luo*, P. Agrawal*, D. Chen, F. Shentu, E. Shelhamer, J. Malik, A. A. Efros, and T. Darrell, “Zero-shot visual imitation,” in *International Conference on Learning Representations (ICLR)*, 2018.

[26] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, “Target-driven visual navigation in indoor scenes using deep reinforcement learning,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[27] G. Li and Y. Yu, “Visual saliency based on multiscale deep features,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[28] G. Li, Y. Xie, L. Lin, and Y. Yu, “Instance-level salient object segmentation,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[29] N. Koenig and A. Howard, “Design and use paradigms for gazebo, an open-source multi-robot simulator,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2004.

[30] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. M. Dollar, “The ycb object and model set: Towards common benchmarks for manipulation research,” in *International Conference on Advanced Robotics (ICAR)*, 2015.

[31] Y. Duan, M. Andrychowicz, B. Stadie, O. J. Ho, J. Schneider, I. Sutskever, P. Abbeel, and W. Zaremba, “One-shot imitation learning,” in *Advances in neural information processing systems (NIPS)*, 2017.

[32] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, “Ros: an open-source robot operating system,” in *ICRA workshop on open source software*, 2009.