Towards Robust Skill Generalization: Unifying Learning from Demonstration and Motion Planning

Muhammad Asif Rana, Mustafa Mukadam, S. Reza Ahmadzadeh, Sonia Chernova, and Byron Boots
Institute for Robotics & Intelligent Machines, Georgia Institute of Technology, Atlanta, GA
{asif.rana, mmukadam3, reza.ahmadzadeh, chernova, bboots}@gatech.edu

Abstract: In this paper, we present *Combined Learning from demonstration And Motion Planning* (*CLAMP*) as an efficient approach to skill learning and generalizable skill reproduction. CLAMP combines the strengths of Learning from Demonstration (LfD) and motion planning into a unifying framework. We carry out probabilistic inference to find trajectories which are *optimal* with respect to a given skill and also *feasible* in different scenarios. We use factor graph optimization to speed up inference. To encode optimality, we provide a new probabilistic skill model based on a stochastic dynamical system. This skill model requires minimal parameter tuning to learn, is suitable to encode skill constraints, and allows efficient inference. Preliminary experimental results showing skill generalization over initial robot state and unforeseen obstacles are presented.

Keywords: Learning from Demonstration, Trajectory Learning, Motion Planning, Probabilistic Inference

1 Introduction

As robots assume collaborative roles alongside humans in dynamic environments, they must have the ability to learn and execute new behaviors to achieve desired tasks. To accomplish this, there are two established approaches for generating trajectories, namely, motion planning [1] and Learning from Demonstration (LfD) [2]. Motion planning focuses on generating trajectories that are optimal with respect to pre-defined criteria (e.g. smooth accelerations) while maintaining feasibility (e.g. obstacle avoidance, reaching via points) [1]. LfD, on the other hand, aims to generate trajectories which satisfy the skill-based constraints learned from demonstrations [2, 3]. As a result, motion planning and LfD can be viewed as having complementary trade-offs. Motion planning generalizes well to new scenarios (comprising the desired/given robot states and the external environment) but requires precise optimality criteria that may be difficult to define for complicated skills, whereas trajectory-based LfD methods circumvent the need for hand coding optimality criteria, but typically do not generalize well.

Our aim is to develop an efficient approach to skill learning and generalizable skill reproduction by combining the strengths of motion planning and trajectory-based LfD while mitigating their weaknesses. Toward this end, we view the problem of generating trajectories as equivalent to probabilistic inference. In this framework, a *posterior* distribution of successful trajectories is computed from a *prior* that encodes optimality, and a *likelihood* that characterizes feasibility in a given scenario. In an earlier application of this view on motion planning [4, 5], the trajectory prior was simple and pre-defined: it encouraged trajectories that minimize acceleration. We argue that the trajectory prior can instead be learned from a set of demonstrations, and our key insight is that the resulting inference based planning paradigm is identical to skill reproduction. The resulting algorithm, Combined Learning from demonstration And Motion Planning (CLAMP), performs probabilistic inference to compute a posterior distribution of trajectories encouraged to match demonstrations while remaining feasible for any given scenario.

Our specific contributions with CLAMP include: (i) a probabilistic skill model (trajectory prior) that extracts the spatio-temporal variance and correlation among the demonstrations in terms of stochastic dynamics, requires minimal parameter tuning while enabling efficient inference; and (ii) a novel

1st Conference on Robot Learning (CoRL 2017), Mountain View, United States.

reproduction method that finds a trajectory via efficient probabilistic inference, which is *optimal* with respect to the learned skill while remaining *feasible* when subjected to different scenarios. We validate our approach on three skills including *box-opening*, *drawer-opening*, and *picking*. We show that CLAMP is capable of successfully reproducing the learned skills and generalizing them over the initial robot state while avoiding new obstacles in the environment.

2 Related Work

Probabilistic methods for trajectory-based LfD provide a viable way to learn a skill from multiple demonstrations. However, the generalization capabilities of these methods vary immensely. Purely probabilistic approaches, including Gaussian mixture models (GMM/GMR) [6] and LfD by Averaging Trajectories (LAT) [7], attract reproduced trajectories towards an average form of the demonstrated motions, without regard to the initial or goal state. Task-parameterized GMM/GMRs [8] generalize better by assigning reference frames to relevant objects and landmarks. Attempts at combining probabilistic approaches with dynamical systems [9, 10, 11] have also met some success at generalization. However, these methods generally require tedious parameter tuning to generate the desired skill models. Although Gaussian processes (GPs) provide a non-parametric alternative [12, 13], the computational complexity of conventional GP approaches scales cubically with the number of data points, limiting their effectiveness in trajectory-based LfD settings.

CLAMP assumes that the demonstrated trajectories are governed by a latent stochastic feedback control policy, which can be approximated as a linear stochastic dynamical system. This simple yet powerful assumption yields a GP over trajectories with an exactly sparse inverse kernel matrix, enabling a significant boost in learning and inference efficiency. This GP produces a Gaussian prior distribution over trajectories. A similar approach, probabilistic movement primitives (ProMPs) [14] directly fits a Gaussian distribution over demonstrations. New skill constraints are incorporated in ProMPs via inference and feedback control policy is then found to follow the resulting trajectory distribution on a robot. In contrast, inference over the prior in CLAMP generates trajectories which naturally follow the demonstrated policy while satisfying all additional constraints.

We consider skill reproduction as performing inference over a prior trajectory distribution, drawing connections to GPMP2 [5], an inference-based planner. Similarly, Dragan et al. [15] show that trajectory adaptation to new start/goal states via dynamic movement primitives (DMPs) [16] is a result of pre-specified Hilbert norm minimization based on finite differences, thus drawing connections to CHOMP [17], a gradient-based trajectory optimizer. The norm minimization procedure in CLAMP, however, goes one step further by minimizing the Mahalanobis distance from a learned prior distribution. While Mukadam et al. [18] describe how GPMP generalizes CHOMP, Dong et al. [5] have shown GPMP2 to provide orders of magnitude faster convergence than CHOMP.

Apart from generalizing skills over different start and goal states, skill reproduction should also generalize to environmental changes, e.g. avoiding unforeseen obstacles. Many conventional LfD approaches are not equipped to handle arbitrarily placed obstacles [6, 7]. Of those that do, obstacle avoidance is rather carried out reactively, without regard to optimality of the entire trajectory [16]. Since motion planning provides a principled way to handle obstacles, attempts at combining LfD and motion planning have been relatively more successful. Ye and Alterovitz [19] presented a hierarchical framework that adapts the output of a learned statistical model to avoid obstacles using a sampling-based motion planner as an ad-hoc post-processing step. However, since the aim of both LfD and motion planning is finding optimal and feasible trajectories, such a hierarchical approach induces redundancies by assuming the two constituent steps to be independent. Recent trajectory optimization based methods take a relatively more unified route. Osa et al. [20] carry out a functional gradient-based optimization for reproduction similar to CHOMP. Not only does their optimization routine carry the same computational inefficiencies as CHOMP (see the discussion in [4]), their demonstration based cost functional disregards the motion dynamics. On the other hand, Koert et al. [21], in a similar approach as ours, carry out probabilistic trajectory optimization. This method performs optimization as a (partially) redundant two-step process. An offline routine first learns a trajectory distribution in the presence of new obstacles and fits a ProMP to represent it, and then an online routine adapts the ProMP given new start/goal states or via-points. A major disadvantage of this approach is that the trajectory distribution has to be re-learned every time an obstacle is displaced or further new obstacles are introduced. In CLAMP, all skill generalization routines are carried out in an efficient one-shot posterior inference procedure, while the trajectory distribution (prior) only encodes human demonstrations.

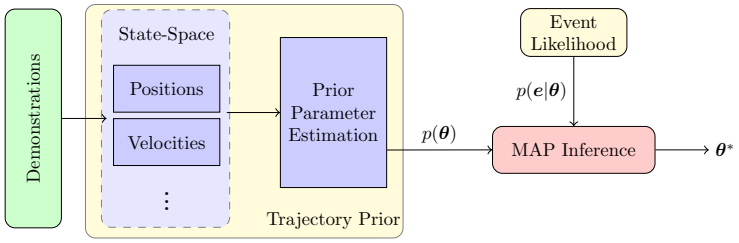


Figure 1: Block diagram showing various components of CLAMP.

3 Trajectory Optimization as Probabilistic Inference

We argue that for generalizable skill reproduction, LfD should adhere to the same motivation as motion planning: finding trajectories that are *optimal* and *feasible*. In contrast to motion planning, where optimality is pre-specified (e.g. smooth accelerations), LfD would require the optimality criteria to be learned from demonstrations. The feasibility criteria represent the reproduction scenario, e.g. collision avoidance, a fixed start state, reaching a desired goal/via-point, or a combination thereof. We adopt the probabilistic inference perspective on motion planning [5, 4] that naturally allows the incorporation of optimality metrics learned from demonstrations in the form of a *prior* distribution. Feasibility is encoded into a *likelihood* function specified in terms of a collection of binary events $\mathbf{e}$. Thus the desired trajectory can be found by calculating the *maximum a posteriori* (MAP) trajectory from the prior and likelihood. Figure 1 illustrates our proposed framework.

3.1 The Trajectory Prior

We define trajectories as continuous-valued functions that map time t to the robot state $\boldsymbol{\theta}(t)$. We define the prior distribution over trajectories with a vector-valued Gaussian process (GP) [22], $\boldsymbol{\theta}(t) \sim \mathcal{GP}(\boldsymbol{\mu}(t), \boldsymbol{\mathcal{K}}(t,t'))$, where $\boldsymbol{\mu}(t)$ is a vector-valued mean function and $\boldsymbol{\mathcal{K}}(t,t')$ is a matrix-valued kernel function. From the definition of a GP, for any collection of time instances $\mathbf{t} \doteq \{t_0, t_1, \ldots, t_N\}$, the corresponding set of vector-valued robot states are jointly Gaussian distributed as

$$\boldsymbol{\theta} \doteq [\boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_N]^T \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\mathcal{K}}), \;\; \boldsymbol{\mu} \doteq [\boldsymbol{\mu}(t_0), \boldsymbol{\mu}(t_1), \ldots, \boldsymbol{\mu}(t_N)]^T, \; \boldsymbol{\mathcal{K}} \doteq [\boldsymbol{\mathcal{K}}(t_i, t_j)]\big|_{ij, 0 \leq i,j \leq N} \quad (1)$$

The support states $\boldsymbol{\theta}_i$, parameterize the continuous-time trajectory at discrete time instances. The GP prior distribution can thus be concisely written as

$$p(\boldsymbol{\theta}) \propto \exp\{-\frac{1}{2}\|\boldsymbol{\theta} - \boldsymbol{\mu}\|^2_{\boldsymbol{\mathcal{K}}}\}. \quad (2)$$

In a motion planning problem, the prior distribution $p(\boldsymbol{\theta})$ defines optimality. Intuitively, this means that during inference, the desired trajectory is encouraged to stay close to the mean of this distribution and is weighted by the kernel. In contrast, we will learn this prior from demonstrations, which can also be interpreted as learning the hyper-parameters (mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\mathcal{K}}$) of the distribution. We employ a sparse, structured GP formulation [23] as detailed in Section 4.

3.2 The Likelihood

The likelihood function specifies information about the new scenarios involved in skill reproduction. Specifically, it is a conditional distribution $\boldsymbol{l}(\boldsymbol{\theta}; \mathbf{e}) \propto p(\mathbf{e}|\boldsymbol{\theta})$ that assigns a probability to the occurrence of random events $\mathbf{e}$, given the trajectory $\boldsymbol{\theta}$. We model the likelihood function as a distribution in the exponential family [4]

$$p(\mathbf{e}|\boldsymbol{\theta}) \propto \exp\{-\frac{1}{2}\|\boldsymbol{h}(\boldsymbol{\theta}; \mathbf{e})\|^2_{\boldsymbol{\Sigma}}\}, \quad (3)$$

where $\boldsymbol{h}(\boldsymbol{\theta}; \mathbf{e})$ is a vector-valued cost function with covariance matrix $\boldsymbol{\Sigma}$. The likelihood defines feasibility for a given trajectory during skill reproduction. In our experiments, we employ events such as collision avoidance and starting from different initial states. The associated distribution and cost functions are detailed in Section 5.

3.3 MAP Inference

The desired optimal and feasible trajectory is the *maximum a posteriori* (MAP) trajectory given the events, i.e. the mode of the posterior distribution $p(\boldsymbol{\theta}|\mathbf{e})$ found through inference,

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left\{p(\boldsymbol{\theta}|\mathbf{e})\right\} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left\{p(\boldsymbol{\theta})p(\mathbf{e}|\boldsymbol{\theta})\right\} \quad (4)$$

In Section 5, we show how to exploit the sparse structure of the problem by representing distributions with factor graphs [24], and use the duality between inference and optimization to provide an efficient approach [5] that solves (4).

4 The Trajectory Prior as a Skill Model

We use Gaussian processes (GPs) to generate the trajectory prior in Section 3.1. Specifically, we employ a special class of structured GPs generated via stochastic differential equations (SDE) [23]. The sparsity in the precision matrix (i.e. inverse covariance matrix) associated with these GPs can be exploited in both learning and inference for efficient computation.

4.1 Structured Heteroscedastic GPs Generated by LTV-SDEs

We view trajectories as solutions to a linear time-varying stochastic differential equation (LTV-SDE)

$$\dot{\boldsymbol{\theta}}(t) = \mathbf{A}(t)\boldsymbol{\theta}(t) + \mathbf{u}(t) + \mathbf{F}(t)\mathbf{w}(t), \quad \mathbf{w}(t) \sim \mathcal{GP}(\mathbf{0}, \mathbf{Q}_C(t)\delta(t-t')), \tag{5}$$

where $\boldsymbol{\theta}(t)$ is the instantaneous robot state consisting of vectorized current positions and their higher-order time derivatives (for all degrees of freedom), $\mathbf{u}(t)$ is a bias term, $\mathbf{A}(t)$ and $\mathbf{F}(t)$ are time-varying system matrices and $\mathbf{w}(t)$ is a white noise process with covariance $\mathbf{Q}_C(t)$ and dirac-delta δ. A similar dynamical system has been employed to generate trajectory distributions in mapping and estimation [23, 25], planning [4, 26], planning and estimation [27], and planning and control [28] problems. However, the key difference here is that the covariance $\mathbf{Q}_C(t)$, is time-varying and hence generates a heteroscedastic GP, which is suitable for encoding the different ways of executing a skill.

Taking the first and second moments of the solution to the LTV-SDE yields the desired GP with

$$\boldsymbol{\mu}(t) = \boldsymbol{\Phi}(t, t_0)\boldsymbol{\mu}_0 + \int_{t_0}^{t} \boldsymbol{\Phi}(t, s)\mathbf{u}(s)ds, \tag{6}$$

$$\boldsymbol{\mathcal{K}}(t, t') = \boldsymbol{\Phi}(t, t_0)\mathbf{Q}_0\boldsymbol{\Phi}(t', t_0)^T + \int_{t_0}^{\min(t,t')} \boldsymbol{\Phi}(t, s)\mathbf{F}(s)\mathbf{Q}_C(s)\mathbf{F}(s)^T\boldsymbol{\Phi}(t', s)^T ds \tag{7}$$

where $\boldsymbol{\Phi}(t, s)$ is the state transition matrix, and $\boldsymbol{\mu}_0$ and $\mathbf{Q}_0$ are the initial mean and covariance. Following Anderson et al. [23], we can decompose the mean, covariance and precision (i.e the inverse covariance) of the GP parameterized by a finite number of support states $\boldsymbol{\theta} = [\boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_N]^T$ as

$$\boldsymbol{\mu} = \mathbf{A}\mathbf{u}, \quad \boldsymbol{\mathcal{K}} = \mathbf{A}\mathbf{Q}\mathbf{A}^T, \quad \boldsymbol{\mathcal{K}}^{-1} = \mathbf{A}^{-T}\mathbf{Q}^{-1}\mathbf{A}^{-1}, \tag{8}$$

where,

$$\boldsymbol{\mu} = \left[\boldsymbol{\mu}(t_0), \boldsymbol{\mu}(t_1), \ldots \boldsymbol{\mu}(t_N)\right]^T, \quad \mathbf{u} = \left[\boldsymbol{\mu}_0, \mathbf{u}_{0,1}, \ldots \mathbf{u}_{N-1,N}\right]^T, \quad \mathbf{u}_{i,i+1} = \int_{t_i}^{t_{i+1}} \boldsymbol{\Phi}(t_{i+1}, s)\mathbf{u}(s)ds,$$

$$\mathbf{Q} = \text{diag}(\mathbf{Q}_0, \mathbf{Q}_{0,1}, \ldots, \mathbf{Q}_{N-1,N}), \qquad \mathbf{Q}_{i,i+1} = \int_{t_i}^{t_{i+1}} \boldsymbol{\Phi}(t_{i+1}, s)\mathbf{F}(s)\mathbf{Q}_C(s)\mathbf{F}(s)^T\boldsymbol{\Phi}(t_{i+1}, s)^T ds,$$

$$\mathbf{A} = \begin{bmatrix} \mathbf{1} & \mathbf{0} & \ldots & \mathbf{0} & \mathbf{0} \\ \boldsymbol{\Phi}(t_1, t_0) & \mathbf{1} & \ldots & \mathbf{0} & \mathbf{0} \\ \boldsymbol{\Phi}(t_2, t_0) & \boldsymbol{\Phi}(t_2, t_1) & \ddots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \mathbf{0} & \mathbf{0} \\ \boldsymbol{\Phi}(t_{N-1}, t_0) & \boldsymbol{\Phi}(t_{N-1}, t_1) & \ldots & \mathbf{1} & \mathbf{0} \\ \boldsymbol{\Phi}(t_N, t_0) & \boldsymbol{\Phi}(t_N, t_1) & \ldots & \boldsymbol{\Phi}(t_N, t_{N-1}) & \mathbf{1} \end{bmatrix}.$$

Due to the lower-triangular form of $\mathbf{A}$, and the block-diagonal form of $\mathbf{Q}$, the precision matrix $\boldsymbol{\mathcal{K}}^{-1}$ has a block-tridiagonal structure. In Section 5, we show how to perform fast and efficient inference by exploiting the exactly sparse structure of this precision matrix.

Note that, from now onwards, $\boldsymbol{\theta}(t)$ will specifically refer to the robot state in configuration space.

4.2 A Combined Prior

Usually, only the demonstrated workspace trajectories are relevant for skill execution. Therefore, we choose to learn a prior distribution $p(\boldsymbol{x}|\boldsymbol{\theta})$ from demonstrations, generated by using the LTV-SDE described in Section 4.1, but defined over the state in workspace $\boldsymbol{x}(t)$, instead of that in configuration space $\boldsymbol{\theta}(t)$. We can directly use $p(\boldsymbol{x}|\boldsymbol{\theta})$ as the prior in (4) to generate a MAP trajectory in workspace.

However, the problem of finding an associated configuration space trajectory is under-constrained for high-degree-of-freedom robots. To resolve this, we introduce a pre-specified smoothness prior in configuration space, $p(\boldsymbol{\theta}) \propto \exp\{-\frac{1}{2}||\boldsymbol{\theta} - \boldsymbol{\mu}^{\boldsymbol{\theta}}||^2_{\boldsymbol{\mathcal{K}}^{\boldsymbol{\theta}}}\}$, giving a combined configuration space prior

$$p_{\boldsymbol{x}}(\boldsymbol{\theta}) = p(\boldsymbol{\theta}|\boldsymbol{x}) \propto p(\boldsymbol{\theta})p(\boldsymbol{x}|\boldsymbol{\theta}). \tag{9}$$

The combined prior eventually functions as our skill model instead of (2). The effect of the combined prior is to yield trajectories that are similar to the demonstrations given in workspace while at the same time maintaining smoothness in configuration space. We detail the procedure for learning the workspace prior $p(\boldsymbol{x}|\boldsymbol{\theta})$, in Section 4.3. The configuration space smoothness prior given by $p(\boldsymbol{\theta})$, is analogous to the homoscedastic GP prior used for motion planning in [4].

Based on the skills required, we may instead choose to directly learn the prior $p(\boldsymbol{\theta})$ in the configuration space, by considering the configuration space demonstrations. In fact, any combination of learned or hand-coded priors in configuration or workspace can be used, as the skill dictates.

4.3 Learned Workspace Prior

The workspace prior distribution in (9) is defined as

$$p(\boldsymbol{x}|\boldsymbol{\theta}) \propto \exp\{-\frac{1}{2}||\mathbf{C}(\boldsymbol{\theta}) - \boldsymbol{\mu}^{\boldsymbol{x}}||^2_{\boldsymbol{\mathcal{K}}^{\boldsymbol{x}}}\}, \tag{10}$$

where the function $\mathbf{C}$ maps a trajectory in configuration space to a workspace trajectory, and the hyper-parameters $\boldsymbol{\mu}^{\boldsymbol{x}}$ and $\boldsymbol{\mathcal{K}}^{\boldsymbol{x}}$ are the mean and the covariance of the distribution. We seek to estimate these hyper-parameters from provided workspace demonstrations.

Since demonstrations are recorded at discrete time instances, we only have access to the support states $\boldsymbol{x}_i$ to estimate the underlying workspace LTV-SDE. A discrete version of the LTV-SDE in (5) proved sufficient for the experiments we considered in this work, defined as

$$\boldsymbol{x}_{i+1} = \boldsymbol{\Phi}^{\boldsymbol{x}}(t_{i+1}, t_i)\boldsymbol{x}_i + \mathbf{u}^{\boldsymbol{x}}_{i,i+1} + \mathbf{w}^{\boldsymbol{x}}_{i,i+1}, \quad \mathbf{w}^{\boldsymbol{x}}_{i,i+1} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}^{\boldsymbol{x}}_{i,i+1}), \tag{11}$$

where the unknown parameters $\boldsymbol{\Phi}^{\boldsymbol{x}}(t_{i+1}, t_i)$, $\mathbf{u}^{\boldsymbol{x}}_{i,i+1}$ and $\mathbf{Q}^{\boldsymbol{x}}_{i,i+1}$ are defined as in (8), but in workspace. Given a set of M trajectory demonstrations $\mathbf{X} = \{\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^M\}$, the regularized maximum likelihood estimate of the unknown parameters for the time interval $[t_i, t_{i+1}]$ is given by

$$\boldsymbol{\Phi}^{\boldsymbol{x}}(t_{i+1}, t_i),\ \mathbf{u}^{\boldsymbol{x}}_{i,i+1} = \underset{\mathbf{u}^{\boldsymbol{x}}_{i,i+1},\ \boldsymbol{\Phi}^{\boldsymbol{x}}(t_{i+1},t_i)}{\operatorname{argmin}} \sum_{m=1}^{M} \|\mathbf{r}^m_{i,i+1}\|^2 + \lambda\|\boldsymbol{\Phi}^{\boldsymbol{x}}(t_{i+1}, t_i)\|^2_F, \tag{12}$$

$$\mathbf{Q}^{\boldsymbol{x}}_{i,i+1} = \frac{1}{M}\sum_{m=1}^{M} \mathbf{r}^m_{i,i+1}{\mathbf{r}^m_{i,i+1}}^T, \tag{13}$$

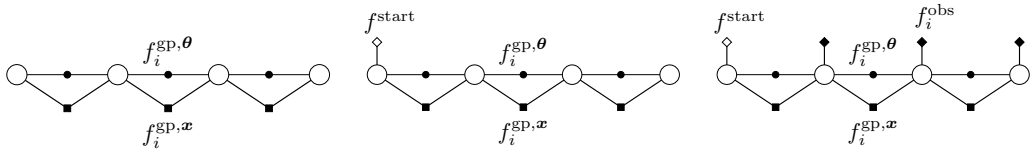
where the residual $\mathbf{r}^m_{i,i+1} = \mathbf{u}^{\boldsymbol{x}}_{i,i+1} - \boldsymbol{x}^m_{i+1} + \boldsymbol{\Phi}^{\boldsymbol{x}}(t_{i+1}, t_i)\boldsymbol{x}^m_i$ and λ is the regularization parameter. We use linear ridge regression [29] to solve (12). The hyper-parameters of the prior are calculated using the relationships in (8). The computational complexity associated with learning the workspace prior is $O(N \cdot d^3)$, where N is the number of support states comprising the trajectory and d is the dimensionality of each support state. Note that, if necessary, a continuous formulation could be learned through variational inference [30].

5 Efficient Inference via Factor Graphs

In this section, we show how to exploit the sparsity of the underlying system to efficiently carry out MAP inference using the learned prior, to reproduce the skill. Any probability distribution P can be represented as a product of functions organized as a bipartite *factor graph* [24] $G = \{\boldsymbol{\Theta}, \mathcal{F}, \mathcal{E}\}$,

$$P(\boldsymbol{\Theta}) \propto \prod_i f_i(\boldsymbol{\Theta}_i), \tag{14}$$

where a set of random variables $\boldsymbol{\Theta} \doteq \{\boldsymbol{\theta}_i\}$ and a set of *factors* $\mathcal{F} \doteq \{f_i\}$ which are functions on variable subsets $\boldsymbol{\Theta}_i$, are connected by a set of edges $\mathcal{E}$. The structure of the precision matrix of a distribution is captured by the structure of its factor graph, i.e. a sparser precision matrix leads to a more factorized distribution. Efficiency during inference is a direct result of this factorization.



(a) (b) (c)

Figure 2: Example factor graphs of (a) the prior distribution, and the joint distribution of the prior and the likelihood when the likelihood describes events associated with (b) different start conditions or (c) obstacle avoidance and different start conditions. States $\boldsymbol{\theta}_i$ are shown as white circles.

5.1 Prior Factors

Using the structured GP formulation (Section 4.1), the combined prior in (9) can be factored as

$$p_{\boldsymbol{x}}(\boldsymbol{\theta}) \propto p(\boldsymbol{\theta})p(\boldsymbol{x}|\boldsymbol{\theta}) \propto f^{\text{gp},\boldsymbol{\theta}} f^{\text{gp},\boldsymbol{x}} = \prod_{i=0}^{N-1} f_i^{\text{gp},\boldsymbol{\theta}}(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{i+1})\ f_i^{\text{gp},\boldsymbol{x}}(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{i+1}), \tag{15}$$

where,

$$f_i^{\text{gp},\boldsymbol{x}}(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{i+1}) = \exp\left\{-\frac{1}{2}\|\boldsymbol{\Phi}^{\boldsymbol{x}}(t_{i+1}, t_i)\mathbf{C}(\boldsymbol{\theta}_i) - \mathbf{C}(\boldsymbol{\theta}_{i+1}) + \mathbf{u}_{i,i+1}^{\boldsymbol{x}}\|^2_{\mathbf{Q}^{\boldsymbol{x}}_{i,i+1}}\right\}, \tag{16}$$

are the workspace prior factors learned from demonstrations as described in Section 4.3, and

$$f_i^{\text{gp},\boldsymbol{\theta}}(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{i+1}) = \exp\left\{-\frac{1}{2}\|\boldsymbol{\Phi}^{\boldsymbol{\theta}}(t_{i+1}, t_i)\boldsymbol{\theta}_i - \boldsymbol{\theta}_{i+1} + \mathbf{u}_{i,i+1}^{\boldsymbol{\theta}}\|^2_{\mathbf{Q}^{\boldsymbol{\theta}}_{i,i+1}}\right\}, \tag{17}$$

are the pre-specified smoothness prior factors in configuration space (see Section 4.2) as described in [4].

5.2 Likelihood Factors

The factorization of the likelihood (Section 3.2) is problem-specific and depends on the events being considered. In this work. we only consider events involving different start conditions and/or collision avoidance. Figure 2(b) shows the joint distribution of the prior and *start-state* likelihood. The posterior inference involves conditioning the prior on a desired start state,

$$p(\mathbf{e}|\boldsymbol{\theta}) \propto f^{\text{start}} = \exp\left\{-\frac{1}{2}\|\boldsymbol{\theta}_0 - \boldsymbol{\theta}_{start}\|^2_{\boldsymbol{\sigma}_{start}}\right\}, \tag{18}$$

where a very small covariance $\boldsymbol{\sigma}_{start}$ signifies the certainty of finding a solution that starts from a desired start state $\boldsymbol{\theta}_{start}$. Figure 2(c) shows the joint distribution with an additional *collision-free* likelihood. The posterior and associated likelihood are then defined as,

$$p(\mathbf{e}|\boldsymbol{\theta}) \propto f^{\text{start}} f^{\text{obs}} = f^{\text{start}}(\boldsymbol{\theta}_0) \prod_{i=1}^{N} f_i^{\text{obs}}(\boldsymbol{\theta}_i), \quad f_i^{\text{obs}}(\boldsymbol{\theta}_i) = \exp\left\{-\frac{1}{2}\parallel \mathbf{h}(\boldsymbol{\theta}_i) \parallel^2_{\boldsymbol{\sigma}_{obs}}\right\}, \tag{19}$$

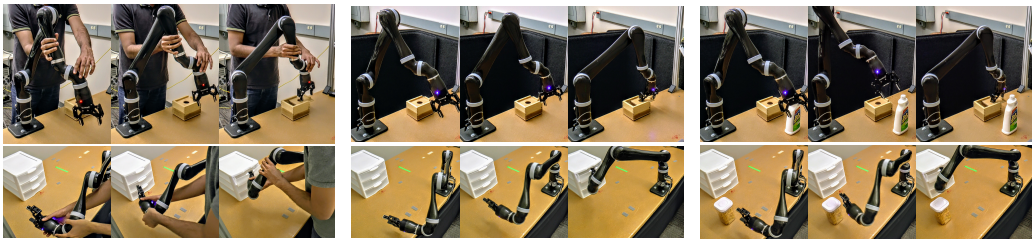
where f_i^{obs} are unary obstacle factors. The collision for any state is evaluated with a precomputed signed distance field, a cost function $\mathbf{h}$, and a hyperparameter $\boldsymbol{\sigma}_{obs}$ that balances the weight on collision avoidance versus staying close to the prior. This technique is also used in GPMP2 for collision avoidance during motion planning (see [5] for details). It is worth noting that, due to this generic formulation, the learned skills can be reproduced in any new environment with never-before-seen obstacles as long as a signed distance field is calculated beforehand.

5.3 Efficient Inference

Finally, for efficient MAP inference (Section 3.3), we take the negative log of the posterior distribution $p(\boldsymbol{\theta}|\mathbf{e}) \propto p_{\boldsymbol{x}}(\boldsymbol{\theta})p(\mathbf{e}|\boldsymbol{\theta})$ using the combined prior (9),

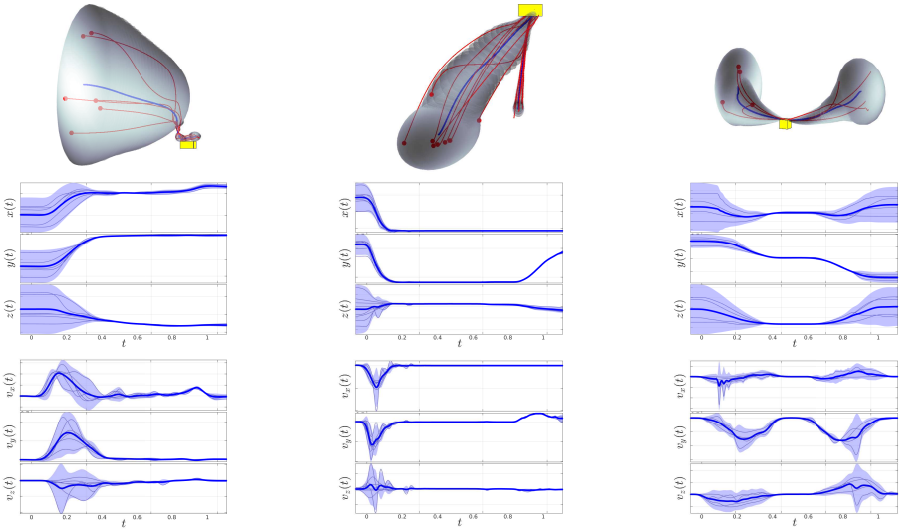
$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\text{argmin}} \left\{\frac{1}{2}\parallel \boldsymbol{\theta} - \boldsymbol{\mu}^{\boldsymbol{\theta}} \parallel^2_{\boldsymbol{\mathcal{K}}^{\boldsymbol{\theta}}} + \frac{1}{2}\parallel \mathbf{C}(\boldsymbol{\theta}) - \boldsymbol{\mu}^{\boldsymbol{x}} \parallel^2_{\boldsymbol{\mathcal{K}}^{\boldsymbol{x}}} + \frac{1}{2}\parallel \boldsymbol{h}(\boldsymbol{\theta}; \boldsymbol{e}) \parallel^2_{\boldsymbol{\Sigma}}, \right\} \tag{20}$$

Thus, giving a nonlinear least squares optimization based formulation for the inference problem. The factor graph allows us to compactly organize the computation, with optimization performed



(a) Kinesthetic demonstration (b) Reproduction (c) Reproduction with an obstacle

Figure 3: Demonstration and reproduction of *box-opening*(top) and *drawer-opening*(bottom)



(a) *box-opening* skill (b) *drawer-opening* skill (c) *picking* skill

Figure 4: Top: Position workspace priors shown in 3D; Middle: Position workspace priors plotted against time; Bottom: Velocity workspace priors plotted againt time. The mean is in blue with an envelope showing the 95% confidence. Demonstrations are overlayed.

using Gauss-Newton or Levenberg-Marquardt. Combining the structure exploiting inference and the quadratic convergence rates of the optimization, make this approach computationally efficient. The computational complexity is directly related to how well the distributions factorize, and since only unary or binary factors are present, the problem is extremely sparse and thus very efficient to solve. The complexity is $O(N \cdot d^3)$, although the complexity would increase in the presence of further higher-order factors that define costs across multiple states.

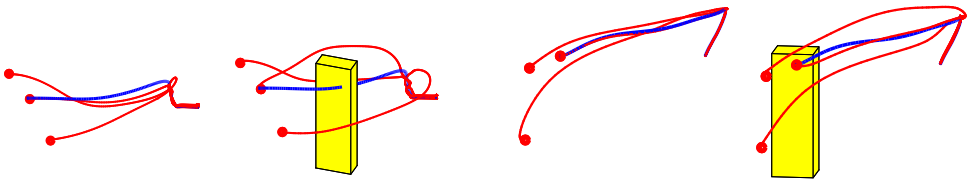
6 Experimental Results

We implemented CLAMP using the GPMP2[1] C++ library and tested it on manipulation problems. For skill learning, we considered workspace state $\boldsymbol{x}(t)$ as composed of end-effector 3D position and linear velocity, which proved sufficient for the experiments considered in this work. We considered joint positions and velocities for the configuration space state $\boldsymbol{\theta}(t)$, and, as in [4], we employed the constant velocity prior for $f^{\text{gp},\boldsymbol{\theta}}$, encouraging smoothness in joint accelerations. The accompanying video shows the experimental results[2].

We validated our proposed method on three skills including, *box-opening*, *drawer-opening* and *picking*. All skills were executed on a Kinova JACO[2] 6-DOF arm. For each skill, we provided multiple demonstrations with different initial end-effector states (varying initial position, zero initial velocity) through kinesthetic teaching [2]. The end-effector positions over time were recorded and the trajectories were temporally aligned using dynamic time warping [31]. The corresponding end-effector linear velocities were estimated by fitting a cubic spline and differentiating with respect to time. Figure 4 shows the learned prior distributions i.e. the skill models.

[1] Available at https://github.com/gtrll/gpmp2

[2] Available at https://youtu.be/DDs_ZxsN0Ek



(a) *box-opening* skill (b) *drawer-opening* skill

Figure 5: Reproduced position trajectories in red from different initial states. The obstacle is in yellow and the prior position mean is in blue.

For the *box-opening* skill, each demonstration is composed of two primitive actions, reaching and sliding the lid of the box. The sliding part of the skill is more constrained compared to the reaching part. As shown in Figure 4(a), the variance in the state variables (i.e. positions and velocities) become much smaller during the sliding portion of the trajectory. For the *drawer-opening* skill, each demonstration involves reaching the drawer handle and pulling it in the direction perpendicular to the drawer body. Like the *box-opening* skill, the second part of the demonstrations are highly restrictive in both positions and velocities to satisfy skill completion, as shown in Figure 4(b). Finally, the *picking* skill involves reaching an object from different initial end-effector positions and then placing it at different locations. As shown in Figure 4(c), since object location is fixed across all demonstrations, the variance in the position state variable is much smaller in the middle part of the skill. However, compared to the other two skills which deal with articulated object manipulation, the velocity profile is not as critical for the *picking* skill. For all the skills, the prior also encodes the coupling between the state variables. This is a consequence of the underlying LTV-SDE.

Provided the initial state of the robot, the likelihood in (18) was used during inference to find MAP trajectories for skill reproduction. For obstacle avoidance, we further incorporated the likelihood in (19). $\boldsymbol{\sigma}_{obs}$ was set manually to enable the desired clearance of the robot from the obstacle. In general, $\boldsymbol{\sigma}_{obs}$ depends on the size of the robot, desired clearance and the environment itself. The MAP trajectories for all scenarios were found using factor graph optimization to solve (20). A video accompanying this paper shows the experimental results.

Figure 5(a) shows the reproduced trajectories for the *box-opening* skill with three different initial robot states. In the left figure, our method was able to adapt the reaching motion as per the initial state. In the presence of a new obstacle (right figure), our method further adapted the reaching part of the skill around the obstacle. The sliding part of the skill is highly constrained, as encoded in the prior and hence does not allow as much adaptability as the reaching part. Figure 5(b) shows three MAP trajectories for the *drawer-opening* skill with different starting states, with and without a new obstacle. In this case, like the others, the reaching part of the skill adapts with varying initial states and obstacles, but the highly constrained pulling part remains more-or-less unchanged. Apart from the position trajectories, the direction of motion is also highly constrained in the latter part of these skills, so having velocities in our prior played a crucial role. In all cases, the robot was successful at executing the desired skill. We note that placing the obstacle in front of the object being manipulated would cause failure due to the robot's inability to carry out the required pulling or sliding action. To detect such failure cases, we can use the workspace prior in (10) to provide a demonstration-based success likelihood of the MAP trajectory $\boldsymbol{\theta}^*$, as a pre-execution evaluation step.

7 Conclusion

We have presented CLAMP, a novel approach which unifies probabilistic LfD and inference-based planning. Within this approach, we first learn the skill in a non-parametric and efficient manner, modeling the underlying system as a stochastic dynamical system. Next, we carried out fast numerical optimization over factor graphs for efficient inference for generalized skill reproduction. Using this approach, we managed to generate trajectories that are optimal with respect to the learned skill (i.e. the trajectory prior) and feasible with respect to the reproduction scenario composed of various events (i.e. the likelihood). Although in our current implementation, we consider robot trajectories to be comprised of positions and velocities and the events to be made up of robot's current initial state and obstacle clearance, our approach allows incorporation of further higher-order dynamics or event likelihoods. We have provided experimental validation of our approach in learning and generalizing object manipulation skills, even in the presence of new obstacles.

Acknowledgments

This research is supported in part by NSF IIS 1637562, NSF IIS 1607299, NSF NRI 1637758, and ONR N00014-16-1-2844.

References

[1] S. M. LaValle. *Planning algorithms*. Cambridge university press, 2006.

[2] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[3] A. J. Ijspeert, J. Nakanishi, and S. Schaal. Learning attractor landscapes for learning motor primitives. In *Advances in neural information processing systems*, pages 1547–1554, 2003.

[4] M. Mukadam, J. Dong, X. Yan, F. Dellaert, and B. Boots. Continuous-time Gaussian process motion planning via probabilistic inference. *arXiv preprint arXiv:1707.07383*, 2017.

[5] J. Dong, M. Mukadam, F. Dellaert, and B. Boots. Motion planning as probabilistic inference using Gaussian processes and factor graphs. In *Proceedings of Robotics: Science and Systems (RSS-2016)*, 2016.

[6] S. Calinon, F. Guenter, and A. Billard. On learning, representing, and generalizing a task in a humanoid robot. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37(2):286–298, 2007.

[7] B. Reiner, W. Ertel, H. Posenauer, and M. Schneider. Lat: A simple learning from demonstration method. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4436–4441. IEEE, 2014.

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

[9] S. M. Khansari-Zadeh and A. Billard. Learning stable nonlinear dynamical systems with Gaussian mixture models. *IEEE Transactions on Robotics*, 27(5):943–957, 2011.

[10] S. Calinon, I. Sardellitti, and D. G. Caldwell. Learning-based control strategy for safe human-robot interaction exploiting task and robot redundancies. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 249–254. IEEE, 2010.

[11] A. Ude, A. Gams, T. Asfour, and J. Morimoto. Task-specific generalization of discrete and periodic dynamic movement primitives. *IEEE Transactions on Robotics*, 26(5):800–815, 2010.

[12] D. B. Grimes, R. Chalodhorn, and R. P. Rao. Dynamic imitation in a humanoid robot through nonparametric probabilistic inference. In *Robotics: Science and Systems (RSS)*, pages 199–206, 2006.

[13] M. Schneider and W. Ertel. Robot learning by demonstration with local Gaussian process regression. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 255–260. IEEE, 2010.

[14] A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann. Probabilistic movement primitives. In *Advances in neural information processing systems*, pages 2616–2624, 2013.

[15] A. D. Dragan, K. Muelling, J. A. Bagnell, and S. S. Srinivasa. Movement primitives via optimization. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2339–2346. IEEE, 2015.

[16] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal. Learning and generalization of motor skills by learning from demonstration. In *2009 IEEE International Conference on Robotics and Automation(ICRA)*, pages 763–768. IEEE, 2009.

[17] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa. CHOMP: Covariant Hamiltonian optimization for motion planning. *The International Journal of Robotics Research*, 32(9-10):1164–1193, 2013.

[18] M. Mukadam, X. Yan, and B. Boots. Gaussian process motion planning. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9–15, 2016.

[19] G. Ye and R. Alterovitz. Demonstration-guided motion planning. In *International symposium on robotics research (ISRR)*, volume 5, 2011.

[20] T. Osa, A. M. G. Esfahani, R. Stolkin, R. Lioutikov, J. Peters, and G. Neumann. Guiding trajectory optimization by demonstrated distributions. *IEEE Robotics and Automation Letters*, 2(2):819–826, 2017.

[21] D. Koert, G. Maeda, R. Lioutikov, G. Neumann, and J. Peters. Demonstration based trajectory optimization for generalizable robot motions. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pages 515–522. IEEE, 2016.

[22] C. E. Rasmussen and C. K. Williams. *Gaussian processes for machine learning*, volume 1. MIT press Cambridge, 2006.

[23] S. Anderson, T. D. Barfoot, C. H. Tong, and S. Särkkä. Batch nonlinear continuous-time trajectory estimation as exactly sparse Gaussian process regression. *Autonomous Robots*, 39(3):221–238, 2015.

[24] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on information theory*, 47(2):498–519, 2001.

[25] X. Yan, V. Indelman, and B. Boots. Incremental sparse GP regression for continuous-time trajectory estimation and mapping. In *Robotics and Autonomous Systems*, volume 87, pages 120–132, 2017.

[26] E. Huang, M. Mukadam, Z. Liu, and B. Boots. Motion planning with graph-based trajectories and Gaussian process inference. In *Proceedings of the 2017 IEEE Conference on Robotics and Automation (ICRA)*, 2017.

[27] M. Mukadam, J. Dong, F. Dellaert, and B. Boots. Simultaneous trajectory estimation and planning via probabilistic inference. In *Proceedings of Robotics: Science and Systems (RSS)*, 2017.

[28] M. Mukadam, C. A. Cheng, X. Yan, and B. Boots. Approximately optimal continuous-time motion planning and control via probabilistic inference. In *Proceedings of the 2017 IEEE Conference on Robotics and Automation (ICRA)*, 2017.

[29] A. E. Hoerl and R. W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

[30] M. K. Titsias. Variational learning of inducing variables in sparse Gaussian processes. In *AISTATS*, volume 5, pages 567–574, 2009.

[31] S. Salvador and P. Chan. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580, 2007.