A Probabilistic Framework for Constrained Manipulations and Task and Motion Planning under Uncertainty

Jung-Su Ha and Danny Driess and Marc Toussaint

***Abstract*— Logic-Geometric Programming (LGP) is a powerful motion and manipulation planning framework, which represents hierarchical structure using logic rules that describe discrete aspects of problems, e.g., touch, grasp, hit, or push, and solves the resulting smooth trajectory optimization. The expressive power of logic allows LGP for handling complex, large-scale sequential manipulation and tool-use planning problems. In this paper, we extend the LGP formulation to stochastic domains. Based on the control-inference duality, we interpret LGP in a stochastic domain as fitting a mixture of Gaussians to the posterior path distribution, where each logic profile defines a single Gaussian path distribution. The proposed framework enables a robot to prioritize various interaction modes and to acquire interesting behaviors such as contact exploitation for uncertainty reduction, eventually providing a composite control scheme that is reactive to disturbance.**

I. Introduction

Manipulation planning problems often involve two major difficulties, namely high-dimensionality and discontinuous contact dynamics, which prohibit widely-used motion planning algorithms such as sampling-based planning [1], [2] or trajectory optimization [3], [4], [5] from being directly applicable. To handle such difficulties, hybrid approaches have been proposed, where additional variables that explicitly represent discrete aspects of problems are incorporated into optimization and jointly optimized [6], [7], [8]. For example, contact invariant optimization [6] relaxes the discontinuity of contact dynamics and utilizes the additional continuous-valued variable to express contact activity that enforces the trajectory to be consistent with physics. In [7], the additional integer variables describe hybrid contact activities and the resulting Mixed-Integer Programming is solved with optimization algorithms involving branch-and-bound. Logic-Geometric Programming (LGP) [8] adopts logic rules to describe discrete aspects of problems on a higher level than typical contact activities, e.g., touch, hit, push, or more general tool-use. A sequence of these logic states (called a *skeleton*) directly implies contact activities over time, which imposes equality/inequality constraints for smooth trajectory optimization. The expressive power of logic enables LGP to enumerate valuable local optima of the planning problem by searching over the logic space.

In this work, we present a probabilistic framework of such hybrid trajectory optimization by extending LGP to stochastic domains, where the dynamics is described stochastically and the cost function is the expectation over all possible trajectories. The corresponding problems can be formulated as stochastic optimal control (SOC), which gives rise to some important and interesting features. First, when comparing various local optima (plans) with different skeletons, the robustness of the plans should be taken into account. For example, consider a planning problem in Fig. 1 whose objective is to push some object on a table towards target area using a single finger or two fingers. Both plans might incur similar costs in a deterministic sense, but the single-finger push strategy is much less favorable in reality because it is more vulnerable to disturbances and uncertainties. Second, we can observe contact exploitation behaviors. To cope with actuator disturbance, a robot might want to fix some parts of its body, e.g. elbow, on the desk to reduce the uncertainty of end-effector's position. Lastly, depending on the deviation from the plan, a robot might decide whether or not to switch to another plan or even to stay in-between them. The original LGP formulation is only deterministic so, even though it can find various feasible plans, a robot cannot help but choose one plan to execute based on the deterministic path cost. In contrast, the probabilistic framework in this work allows a robot for prioritizing different plans by taking robustness as well as deterministic path cost into account and for acquiring interesting contact exploitation behaviors. Furthermore, a composite reactive controller constructed from various plans enables a robot to adaptively choose which plan(s) to follow.

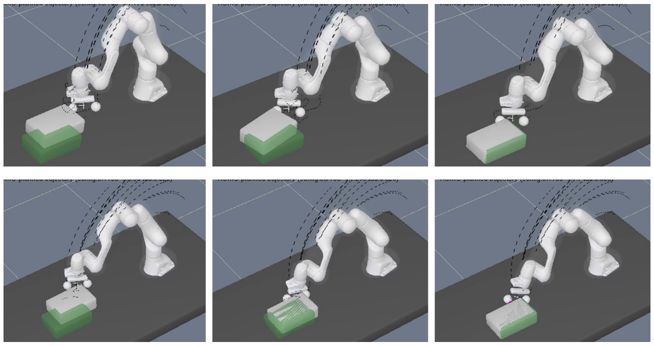


Fig. 1. Two strategies for box pushing. A robot can use one finger (upper), or two fingers (lower). Obviously, the latter strategy is more robust to disturbance, thereby incurring smaller feedback control cost.

The technical aspect of this work is based on the duality between control and inference [9], [10], [11]. Under this duality, motion planning is equivalent to inference of posterior path distribution. As in trajectory optimization, the gradient and Hessian can accelerate the inference procedure, which relates to the Laplace approximation [12]. Given the fact that the prescribed logics provide smoothness of sub-problems, we interpret LGP in stochastic domains as fitting a mixture of Gaussians to the posterior path distribution, where each skeleton defines a single Gaussian path distribution.

All authors are with the Machine Learning & Robotics Lab, University Stuttgart and with the Max Planck Institute for Intelligent Systems, Stuttgart, Germany jung-su.ha@ipvs.uni-stuttgart.de

II. Background

A. Stochastic optimal control as KL-minimization

Consider the configuration space $\mathcal{X} = \mathbb{R}^{d_x}$ of an d_x-dimensional robot and an initial configuration $x_0 \in \mathcal{X}$ and a velocity $\dot{x}_0$ are given. Let $\mathbf{z} = (x, \dot{x}) \in \mathbb{R}^{2d_x}$ be a state vector, $\mathbf{u} \in \mathbb{R}^{d_u}$ be a control vector which represents torques or desired accelerations of actuated joints ($d_u \leq d_x$), and $\mathbf{w}$ be a d_u-dimensional Wiener process that is injected into the robot's actuators. Then the robot dynamics can be written as the following stochastic differential equation (SDE), which is affine in the control input and the disturbance:

$$d\mathbf{z}(t) = \mathbf{f}(\mathbf{z}(t))dt + G(\mathbf{z}(t))(\mathbf{u}(t)dt + \sigma d\mathbf{w}(t)), \quad (1)$$

where $\mathbf{f} : \mathbb{R}^{2d_x} \rightarrow \mathbb{R}^{2d_x}$ is the passive dynamics and $G : \mathbb{R}^{2d_x} \rightarrow \mathbb{R}^{2d_x \times d_u}$ is the control transition matrix function. With an instantaneous state cost rate $V : \mathbb{R}^{2d_x} \rightarrow \mathbb{R}^+$, an SOC problem is formulated as follow:

$$J = \mathbb{E}_{p_\mathbf{u}}\left[V_T(\mathbf{z}(T)) + \int_0^T V(\mathbf{z}(t)) + \frac{1}{2\sigma^2}\mathbf{u}(t)^T\mathbf{u}(t)dt\right], \quad (2)$$

where $p_\mathbf{u}$ is the probability measure induced by the controlled trajectories in (1), with $\mathbf{z}(0) = \mathbf{z}_0 = (x_0, \dot{x}_0)$. The objective of an SOC problem is then to find a control policy $\mathbf{u}(t) = \pi^*(\mathbf{z}(t), t)$ that minimizes the cost functional (2).

The above types of SOC problems, which are defined with control/disturbance-affine dynamics and quadratic control cost, are called linearly solvable optimal control problems and have interesting properties [13], [11]. In particular, they can be transformed into Kullback-Leibler (KL) divergence minimization [14], [15] using the following theorem.

Theorem 1 (Girsanov's Theorem [16]): Suppose $p_\mathbf{0}$ *is the probability measures induced by the uncontrolled trajectories from* (1) *with* $\mathbf{z}(0) = (x_0, \dot{x}_0)$ *and* $\mathbf{u}(t) = 0 \; \forall t \in [0, T]$. *Then, the Radon-Nikodym derivative of* $p_\mathbf{u}$ *with respect to* $p_\mathbf{0}$ *is given by*

$$\frac{dp_\mathbf{u}}{dp_\mathbf{0}} = \exp\left(\frac{1}{2\sigma^2}\int_0^T ||\mathbf{u}(t)||^2 dt + \frac{1}{\sigma}\int_0^T \mathbf{u}(t)^T d\mathbf{w}(t)\right), \quad (3)$$

where $\mathbf{w}(t)$ *is a Wiener process for* $p_\mathbf{u}$.

With Girsanov's theorem, the objective function (2) can be rewritten in terms of KL divergence:

$$\begin{aligned} J &= \mathbb{E}_{p_\mathbf{u}}\left[V_T(\mathbf{z}(T)) + \int_0^T V(\mathbf{z}(t)) + \frac{1}{2\sigma^2}||\mathbf{u}(t)||^2 dt\right] \\ &= \mathbb{E}_{p_\mathbf{u}}\left[V_T(\mathbf{z}(T)) + \int_0^T V(\mathbf{z}(t))dt + \log\frac{dp_\mathbf{u}(\mathbf{z}_{[0,T]})}{dp_\mathbf{0}(\mathbf{z}_{[0,T]})}\right] \\ &= \mathbb{E}_{p_\mathbf{u}}\left[\log\frac{dp_\mathbf{u}(\mathbf{z}_{[0,T]})}{dp_\mathbf{0}(\mathbf{z}_{[0,T]})\exp(-V(\mathbf{z}_{[0,T]}))/\xi} - \log\xi\right] \\ &= D_{\text{KL}}\left(p_\mathbf{u}(\mathbf{z}_{[0,T]})||p^*(\mathbf{z}_{[0,T]})\right) - \log\xi, \quad (4) \end{aligned}$$

where $\mathbf{z}_{[0,T]} \equiv \{\mathbf{z}(t);\ \forall t \in [0, T]\}$ is a state trajectory, $V(\mathbf{z}_{[0,T]}) \equiv V_T(\mathbf{z}(T)) + \int_0^T V(\mathbf{z}(t))dt$ is a trajectory state cost and $\xi \equiv \int \exp(-V(\mathbf{z}_{[0,T]}))dp_\mathbf{0}(\mathbf{z}_{[0,T]})$ is a normalization constant.[1] Because ξ is a constant, $p^*(\mathbf{z}_{[0,T]})$ can be interpreted as the optimally-controlled trajectory distribution that minimizes the cost functional (2):

$$dp^*(\mathbf{z}_{[0,T]}) = \frac{\exp(-V(\mathbf{z}_{[0,T]}))dp_\mathbf{0}(\mathbf{z}_{[0,T]})}{\int \exp(-V(\mathbf{z}_{[0,T]}))dp_\mathbf{0}(\mathbf{z}_{[0,T]})}, \quad (5)$$

and the corresponding optimal cost is given by

$$J^* = -\log\int \exp(-V(\mathbf{z}_{[0,T]}))dp_\mathbf{0}(\mathbf{z}_{[0,T]}). \quad (6)$$

Once the optimal trajectory distribution is obtained, the optimal control can be recovered by enforcing the controlled dynamics to mimic the optimal trajectory distribution, e.g., via moment matching. By applying Girsanov's theorem to (5), the optimal trajectory distribution can be expressed as:

$$dp^*(\mathbf{z}_{[0,T]}) \propto dp_\mathbf{u}(\mathbf{z}_{[0,T]})\exp\left(-V_\mathbf{u}(\mathbf{z}_{[0,T]})\right), \quad (7)$$

where $V_\mathbf{u}(\mathbf{z}_{[0,T]}) = V(\mathbf{z}_{[0,T]}) + \frac{1}{2\sigma^2}\int_0^T ||\mathbf{u}(t)||^2 dt + \int_0^T \mathbf{u}(t)^T d\mathbf{w}(t)$. The SOC framework that utilizes the importance sampling scheme to approximate this distribution is called path integral control [17], [14]. It samples a set of trajectories $\mathbf{z}^l_{[0,T]} \sim p_\mathbf{u}(\cdot)$, assigns their importance weights as $\tilde{w}^l \propto \exp(-V_\mathbf{u}(\mathbf{z}^l_{[0,T]}))$, and computes the optimal control by matching the first (and second) moments of $p_\mathbf{u}$ to p^*.

B. Laplace approximation of path distributions

Instead of relying on sampling schemes for approximating p^*, this work builds on the efficient local optimization methods by investigating a close connection between second order trajectory optimization algorithms [3], [4], [5][2] and the Laplace approximation. The Laplace approximation fits a normal distribution to the first two derivatives of the log target density function at the mode. Let $\mathbf{x} = x_{1:N} = (x_1, x_2, ..., x_N) \in \mathbb{R}^{N \times d_x}$ be a path representation of $\mathbf{z}_{[0,T]}$, which is a path of N time steps in the configuration space $\mathcal{X}$. In this path representation, the velocity and acceleration (and control inputs) of the joints can be computed from two and three consecutive configurations, respectively, using the finite difference approximation.[3] Slightly abusing the notation, the uncontrolled path distribution and the state trajectory cost are then expressed as functions of $\mathbf{x}$:

$$p_\mathbf{0}(\mathbf{x}) \propto \exp\left(-\sum_{n=1}^N f_\mathbf{0}(x_{n-2:n})\right), \quad (8)$$

$$\exp\left(-V(\mathbf{x})\right) = \exp\left(-\sum_{n=1}^N f_V(x_{n-1:n})\right), \quad (9)$$

for an appropriately given prefix $x_{-1:0}$. Then, the problem of finding the mode $\mathbf{x}^*$ of $p^*(\mathbf{x}) \propto p_\mathbf{0}(\mathbf{x})\exp\left(-V(\mathbf{x})\right)$ is an unconstrained nonlinear program (NLP):

$$\min_{x_{1:N}} \sum_{n=1}^N f_\mathbf{0}(x_{n-2:n}) + f_V(x_{n-1:n}), \quad (10)$$

[1] Note that the second term in the exponent of (3) disappears when taking expectation w.r.t. $p_\mathbf{u}$, i.e. $\mathbb{E}_{p_\mathbf{u}}[\int_0^T \mathbf{u}(t)^T d\mathbf{w}(t)] = 0$.

[2] Our method especially builds upon the framework of *k*-order Motion Optimization (KOMO) [5] which has the same efficiency as the others while being able to address more general problems.

[3] The path representation significantly reduces the size of optimization problems, which leads to better numerical stability [18], [19], [5].

which can be solved using the Newton-Raphson algorithm, i.e., $\mathbf{x}^{i+1} = \mathbf{x}^i - \nabla^2 f(\mathbf{x}^i)^{-1} \nabla f(\mathbf{x}^i)$ where $f(\mathbf{x}) = \sum_{n=1}^N f_\mathbf{0}(x_{n-2:n}) + f_V(x_{n-1:n})$. With a solution, $\mathbf{x}^*$, and a Hessian at the solution, $\nabla^2 f(\mathbf{x}^*)$, the resulting Laplace approximation is given by:

$$p^*(\mathbf{x}) \approx \mathcal{N}(\mathbf{x}|\mathbf{x}^*, \nabla^2 f(\mathbf{x}^*)^{-1}). \tag{11}$$

The optimal cost is also approximated similarly from (6):

$$J^* \approx f(\mathbf{x}^*) - \frac{1}{2}\log\frac{|\nabla^2 f_\mathbf{0}(\mathbf{x}^*)|}{|\nabla^2 f(\mathbf{x}^*)|}. \tag{12}$$

Note that the covariance of the approximate distribution has full rank for fully-actuated robots ($d_u = d_x$), but not for underacted robots ($d_u < d_x$). In such cases, the NLP should be formulated with equality constraints that restrict the uncontrollable subspace to be consistent with the dynamics (1). Details will be addressed in Section III-B.

III. Estimating the Path Distribution for Constrained Trajectory Optimization

A. Logic geometric programming in stochastic domains

We now consider more general manipulation planning problems where the configuration space $\mathcal{X} = \mathbb{R}^{d_x} \times SE(3)^m$ involves m rigid objects as well as a d_x-dimensional robot. The dynamics in (1) becomes complicated in this case, because it should express interactions between the robot and the objects. The objects are controllable only when they are in contact with the robot, thus the dynamics (1) is discontinuous around the contact activities. Local optimization methods are no longer effective since they cannot utilize the well-defined gradient and Hessian along the directions of contact switching. Logic Geometric Programming (LGP) addresses this difficulty by augmenting the formulation with additional logic decision variables, $(s_{1:K}, a_{1:K})$, which describe discrete aspects of dynamics in a higher level, e.g., touch, hit, push, or more general tool-use. A mode s_k imposes a set of equality/inequality constraints for the prescribed contact activities to the optimization while a switch a_k represents transitions between the modes. We can formulate an LGP problem in stochastic domains as follows:

$$\min_{\substack{\mathbf{u}_{[0,T]},\\ a_{1:K}, s_{1:K}}} E_{p_\mathbf{u}}\left[V_T(\mathbf{z}(T)) + \int_0^T V(\mathbf{z}(t)) + \frac{1}{2\sigma^2}||\mathbf{u}(t)||^2 dt\right]$$

$$\begin{aligned}
\text{s.t. } \forall t \in [0,T]: \ & h_{\text{path}}(\mathbf{z}(t), s_{k(t)}) = 0, \ g_{\text{path}}(\mathbf{z}(t), s_{k(t)}) \leq 0 \\
& d\mathbf{z}(t) = \mathbf{f}_{s_{k(t)}}(\mathbf{z}(t))dt + G_{s_{k(t)}}(\mathbf{z}(t))(\mathbf{u}(t)dt + \sigma d\mathbf{w}(t)) \\
& \forall_{k=1}^K: \ h_{\text{switch}}(\mathbf{z}(t), a_{k(t)}) = 0, \ g_{\text{switch}}(\mathbf{z}(t), a_{k(t)}) \leq 0, \\
& \qquad s_k \in \text{succ}(s_{k-1}, a_k).
\end{aligned} \tag{13}$$

Here, the SDE is conditioned on s so that it can be defined only in the remaining subspace which is not constrained by the path constraints, $(h, g)_{\text{path}}$. For example, when a mode specifies manipulation of a particular object, the SDE represents the robot's dynamics constrained for that specified interaction and the dynamics of the manipulated object is defined by path constraints. Because the contact activities are prescribed by the *skeleton* $a_{1:K}$ and the smooth switch constraints $(h, g)_{\text{switch}}$, the dynamics in (13) is now smooth w.r.t. the state and the control inputs, thereby making the corresponding SOC, $\mathcal{P}(a_{1:K})$, to be smooth.

LGP problems are often addressed with a two-level hierarchical approach [20], [8], where a higher-level module proposes a skeleton $a_{1:K}$ using, e.g., tree search and a lower-level NLP solver returns a solution of $\mathcal{P}(a_{1:K})$. LGP in stochastic domains (13) has two distinctive features: While $\mathcal{P}(a_{1:K})$ is evaluated for a single optimal trajectory in the deterministic case, the path distribution should instead be considered, which results in the additional stochastic cost term (Sec. III-B); also, considering various modes allows for constructing the composite reactive control law (Sec. IV-B).

B. Probabilistic LGP as fitting a mixture of Gaussians

Let $\{\mathbf{a}_i = a_{1:K}{}^{(i)}; i = 1, ..., N_a\}$ be a set of candidate skeletons for an LGP problem (13). We now attempt to approximate the optimal path distribution as a mixture distribution, where each skeleton defines one mixture component:

$$p^*(\mathbf{x}) \approx \sum\nolimits_{i=1}^{N_a} p^*(\mathbf{a}_i) p^*(\mathbf{x}|\mathbf{a}_i). \tag{14}$$

The mixture component $p^*(\mathbf{x}|\mathbf{a}_i)$ corresponds to the SOC problem $\mathcal{P}(\mathbf{a}_i)$, of which support is defined by the constraints of the original problem, i.e.,:

$$\begin{aligned}
& p^*(\mathbf{x}|\mathbf{a}_i) \propto p_\mathbf{0}(\mathbf{x}|\mathbf{s}_i)\exp(-V(\mathbf{x})) \\
& \text{s.t. } h_{\text{path}}(\mathbf{x}, \mathbf{s}) = 0, \ g_{\text{path}}(\mathbf{x}, \mathbf{s}) \leq 0, \\
& \quad h_{\text{switch}}(\mathbf{x}, \mathbf{a}) = 0, \ g_{\text{switch}}(\mathbf{x}, \mathbf{a}) \leq 0, \\
& \quad \forall_{k=1}^K: \ s_k \in \text{succ}(s_{k-1}, a_k).
\end{aligned} \tag{15}$$

Given that $\mathcal{P}(a_{1:K})$ is a *smooth* SOC problem, we can use the Laplace approximation to represent each mixture component $p^*(\mathbf{x}|\mathbf{a}_i)$ as a Gaussian distribution. Apart from the unconstrained NLP in (10), however, $\mathcal{P}(a_{1:K})$ yields a more complicated trajectory optimization problem; e.g., the dynamics of moving or manipulated objects are defined by equality constraints and the resting objects are just imposing inequality constraints for collision avoidance. Such problems should be formulated as a constrained NLP:

$$\begin{aligned}
& \min_{x_{1:N}} \sum_{n=1}^N f_\mathbf{0}(x_{n-2:n}) + f_V(x_{n-1:n}) \\
& \text{s.t. } \forall_{n=1}^N: h(x_{n-1:n}) = 0, \ g(x_{n-1:n}) \leq 0,
\end{aligned} \tag{16}$$

which can be addressed by any constrained optimization methods, such as Augmented Lagrangian Gauss-Newton.

Suppose we have found $\mathbf{x}_i^*$, an NLP solution for the i^{th} skeleton. We then approximate the i^{th} mixture density as:

$$p^*(\mathbf{x}|\mathbf{a}_i) \approx \mathcal{N}(\mathbf{x}|\mathbf{x}_i^*, \Sigma_i^*). \tag{17}$$

Because of the equality/inequality constraints imposed in (16), this distribution is degenerate; i.e., deviations from $\mathbf{x}_i^*$ can only lie in the kernel of the equality and active inequality constraint Jacobians, thereby making the above distribution only span a lower-dimensional subspace. Let a column of matrix W denote an orthonormal basis of the nullspace of $J = \begin{bmatrix} \nabla h(\mathbf{x}^*) \\ \nabla \text{diag}(\lambda) g(\mathbf{x}^*) \end{bmatrix}$, where λ is the dual variables on the inequality constraints. Then, Σ_i^* is given by the inverse of the projected second derivatives of $\log p^*(\mathbf{x})$ at $\mathbf{x}^*$:

$$\Sigma_i^* = W \left(W^T \nabla^2 f(\mathbf{x}_i^*) W\right)^{-1} W^T. \tag{18}$$

To complete the mixture approximation, we also need to compute the mixture weights, $p^*(\mathbf{a}_i)$. Because a skeleton imposes different constraints to the corresponding NLP (16), making each mixture component span different subspaces, we can assume that the modes are widely separated, which enables the mixture weights to be computed independently [21, Chapter 12]:

$$\begin{aligned} p^*(\mathbf{a}_i) &\propto p_{\mathbf{0}}(\mathbf{x}_i^*|\mathbf{s}_i) \exp(-V(\mathbf{x}_i^*)) \{(2\pi)^d |\Sigma_i^*|_+\}^{1/2} \\ &\propto \exp\left(-f(\mathbf{x}_i^*)\right) \left(|\Sigma_i^*|_+ / |\Sigma_i|_+\right)^{1/2}, \end{aligned} \tag{19}$$

where $\Sigma_i = W \left(W^T \nabla^2 f_{\mathbf{0}}(\mathbf{x}_i^*) W\right)^{-1} W^T$ is the covariance of the (degenerate) uncontrolled path distribution $p_{\mathbf{0}}(\mathbf{x}|\mathbf{a}_i)$, $d = \text{rank}(\Sigma_i^*)$, and $|\cdot|_+$ denotes a pseudo-determinant. Note that the mixture weights (19) are determined by two factors. The first term is a cost of the optimal deterministic path which are exponentially penalized. The second term can be interpreted as a stochastic cost that penalizes an entropy ratio between the optimal and uncontrolled path distributions; robust plans have large margins for deviations from the reference so the controlled path distribution need not be shrunk via feedback control, while plans vulnerable to disturbance do not allow even a small deviation, requiring a high-gain feedback controller to make the controlled path distribution relatively narrow. In other words, this term represents the expected *feedback* control cost from the optimal controller. See the equivalence between the optimal value of the unimodal approximation (12) and the log of the mixture weights (19); the mass of each mixture component is assigned according to the total (deterministic + stochastic) cost incurred by that plan. Finally, the multimodal approximation of the optimal cost is given by:

$$J^* \approx -\log \sum_{i=1}^{N_a} \frac{1}{N_a} \exp\left(-f(\mathbf{x}_i^*)\right) \frac{|\Sigma_i^*|_+^{1/2}}{|\Sigma_i|_+^{1/2}}, \tag{20}$$

which, of course, becomes the cost in (12) when $N_a = 1$.

IV. Reactive Controller for Mode Switching

After planning, we need a control policy to execute the plan. Using the multi-modal path distribution, the control policy should be able to not only stabilize a particular reference trajectory, but also decide which reference trajectory to follow. This section is devoted to derive the stabilizing controllers within each mode (Sec. IV-A) and to introduce two methods to synthesize those controllers (Sec. IV-B).

A. *k-order constrained dynamic programming*

Within each mode, to derive the controller for general constrained cases, consider the following recursive equation:

$$\begin{aligned} &J_n(x_{n-2:n-1}) \\ &= \min_{x_{n:N}} \sum_{l=n}^{N} f_l(x_{l-2:l}) \text{ s.t. } \forall_{l=n}^{N} : h_l = 0,\ g_l \leq 0 \\ &= \min_{x_n} \left[f_n(x_{n-2:n}) + J_{n+1}(x_{n-1:n})\right] \text{ s.t. } h_n = 0,\ g_n \leq 0, \end{aligned} \tag{21}$$

where $J_{N+1} \equiv 0$. Such procedures for computing the cost-to-go function J_n is called k-order constrained dynamic programming (KODP) [5, Sec. 4], which is an generalization of the Bellman optimality equation to the k-order (2nd-order in (21)) and constrained cases.

In particular, the linear feedback controller for a computed plan can be built directly from the gradient and Hessian of the optimization. Let $\delta x = x - x^*$ and consider the quadratic and linear approximations of J, f, h, and g:

$$\begin{aligned} J_n(x_{n-2:n-1}) &\equiv \frac{1}{2} \delta x^T V_n \delta x + v_n^T \delta x + \bar{v} \\ f_n(x_{n-2:n-1}) &\approx \frac{1}{2} \delta x^T \nabla^2 f^* \delta x + (\nabla f^*)^T \delta x + f^* \\ h_n(x_{n-2:n-1}) &\approx (\nabla h^*)^T \delta x,\ g_n(x_{n-2:n-1}) \approx (\nabla g^*)^T \delta x. \end{aligned}$$

Note that all the gradients (and Hessian) of f, h, g are already computed while solving the NLP (16).[4] The minimization in KODP (21) is then written as:

$$\begin{aligned} \delta x_n^* = &\operatorname*{argmin}_{x_n} f(\delta x_{n-2:n}) + J_{n+1}(\delta x_{n-1:n}) \\ &\text{s.t. } h_n(\delta x_{n-2:n}) = 0,\ g_n(\delta x_{n-2:n}) \leq 0, \end{aligned} \tag{22}$$

which has the form of a quadratic program (QP) with[5]

$$\begin{aligned} f_n + J_{n+1} \equiv \frac{1}{2} &\begin{bmatrix} \delta x_{n-2:n-1} \\ \delta x_n \end{bmatrix}^T \begin{bmatrix} D_n & C_n \\ C_n^T & E_n \end{bmatrix} \begin{bmatrix} \delta x_{n-2:n-1} \\ \delta x_n \end{bmatrix} \\ &+ \begin{bmatrix} d_n \\ e_n \end{bmatrix} \begin{bmatrix} \delta x_{n-2:n-1} \\ \delta x_n \end{bmatrix} + c_n, \\ \text{s.t. } &\begin{bmatrix} l_n \\ m_n \end{bmatrix}^T \begin{bmatrix} \delta x_{n-2:n-1} \\ \delta x_n \end{bmatrix} = 0. \end{aligned} \tag{23}$$

Given the cost-to-go function at the next time step J_{n+1}, the solution of the above QP can be represented linearly around $\delta x = 0$ (which corresponds to the solution of the original problem) using the sensitivity analysis of NLPs [22], [23]:

$$\begin{bmatrix} E_n & m_n \\ m_n^T & 0 \end{bmatrix} \begin{bmatrix} \delta x_n^* \\ \delta \lambda_n^* \end{bmatrix} = \begin{bmatrix} -C_n^T \delta x_{n-2:n-1} - e_n \\ -l_n^T \delta x_{n-2:n-1} \end{bmatrix}. \tag{24}$$

The above directly implies the linear feedback control law:

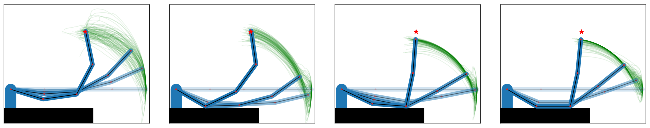
$$\begin{aligned} \begin{bmatrix} \delta x_n^* \\ \delta \lambda_n^* \end{bmatrix} &= -\begin{bmatrix} E_n & m_n \\ m_n^T & 0 \end{bmatrix}^{-1} \left(\begin{bmatrix} e_n \\ 0 \end{bmatrix} + \begin{bmatrix} C_n^T \\ l_n^T \end{bmatrix} \delta x_{n-2:n-1} \right) \\ &= u_n^{ff} + K_n \delta x_{n-2:n-1}. \end{aligned} \tag{25}$$

The cost-to-go functions along the whole time horizon can be derived from the Bellman equation $J_n = \min_{\delta x_n} \left[f_n + J_{n+1}\right]$ which results in the following backward matrix recursion:

$$\begin{aligned} V_n = &D_n + \frac{1}{2} \begin{bmatrix} C_n & l_n \end{bmatrix} \bar{H}_n \begin{bmatrix} C_n^T \\ l_n^T \end{bmatrix} - \begin{bmatrix} C_n & l_n \end{bmatrix} H_n \begin{bmatrix} C_n^T \\ 0 \end{bmatrix} \\ v_n = &d_n - \begin{bmatrix} C_n & 0 \end{bmatrix} H_n \begin{bmatrix} e_n \\ 0 \end{bmatrix} + \begin{bmatrix} C_n & l_n \end{bmatrix} \left(\bar{H}_n - H_n\right) \begin{bmatrix} e_n \\ 0 \end{bmatrix} \\ \bar{v}_n = &c_n + \frac{1}{2} \begin{bmatrix} e_n^T & 0 \end{bmatrix} \left(\bar{H}_n - 2H_n\right) \begin{bmatrix} e_n \\ 0 \end{bmatrix}, \end{aligned} \tag{26}$$

[4] We can also include additional penalties like $f \leftarrow f + \rho||x - x^*||^*$ or modify the weights between cost terms to adjust the closed loop behaviors.

[5] We leave out the inequality constraints for the sake of notation but the activated inequalities should be treated as the equality constraints.



(a) Free Reaching (b) Fix Joint1 (c) Fix Joint2 (d) Fix Joint1&2

Fig. 2. The snapshots of the optimized path for the four different strategies and sampled end-effector trajectories from the optimal path distributions.

where $H_n \equiv \begin{bmatrix} E_n & m_n \\ m_n^T & 0 \end{bmatrix}^{-1}$ and $\bar{H}_n \equiv H_n \begin{bmatrix} E_n & 0 \\ 0 & 0 \end{bmatrix} H_n$. Note that, in the 1st-order unconstrained case, the above recursion (26) is equivalent to the Riccati equation of LQR.

B. Composite optimal control policy

During the execution, the mixture weights can be updated *on the fly*. Let $J_n^{(i)}$ be the cost-to-go function w.r.t. an ith skeleton at a time step $l = n$ and $\tilde{\Sigma}$ be a covariance matrix that is only for the future trajectory $l = n, ..., N$ which only takes submatrix of $\nabla^2 f(\mathbf{x}_i^*)$ or $\nabla^2 f_{\mathbf{0}}(\mathbf{x}_i^*)$. Then, the mixture weight of an ith skeleton is given as:

$$p_n^*(\mathbf{a}_i) \propto \exp\left(-J_n^{(i)}(x_{n-2:n-1})\right)\left(|\tilde{\Sigma}_i^*|_+/|\tilde{\Sigma}_i|_+\right)^{1/2} \quad (27)$$

where $x_{n-2:n-1}$ is the two past configurations that the robot observed. With these mixture weights, we introduce two different methods to construct the composite control policies from all skeletons, $u_n^{(i)}$ in (25).

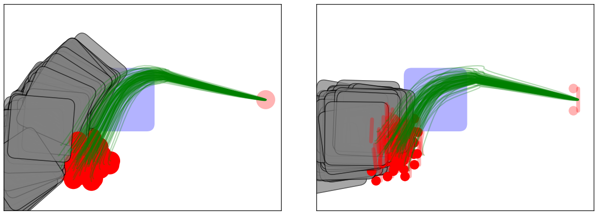
- **Blending**: As suggested in [24], [25], the control input can be computed as a linear combination, i.e.,
$$u_n^* = \sum\nolimits_i p_n^*(\mathbf{a}_i) u_n^{(i)}, \quad (28)$$
which minimizes forward KL divergence $D_{\mathrm{KL}}(p^* || p_{\mathbf{u}})$.
- **Switching**: The blending method can cause undesirable smoothing effects in practice because it mixes behaviors for different contact activities. An alternative is to simply take the best expected policy as:
$$u_n^* = u_n^{(i^*)}, \ i^* = \operatorname*{argmax}_i p^*(\mathbf{a}_i). \quad (29)$$

We briefly show the different resulting behaviors of two methods in the following section.

The overall framework for planning and control can be summarized as follows: (i) In the offline planning phase, trajectories w.r.t. the different candidate skeletons proposed by a logic-level planner are optimized by solving (16), and for each skeleton, the cost-to-go function as well as the linear feedback policy are computed from KODP (26) and (25). (ii) In the online execution phase, the mixture weights of the skeletons are assigned as (27) and, based on those, the control input is computed via (28) or (29). By considering various candidate plans and appropriately building composite policies, a robot can not only choose a more efficient and robust plan but also flexibly react to disturbance (e.g., stay in the current plan or switch to another).

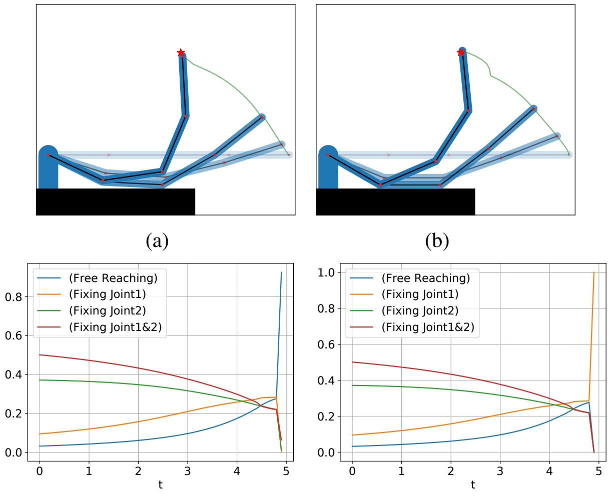
V. Demonstration

We demonstrate our approach on three manipulation planning problems, reaching a target, pushing an object, and touching a banana. For clearer visualization and more results, we refer the readers to the accompanying video.



(a) Single-finger push (b) Two-finger push

Fig. 3. Sample trajectories from $\mathcal{N}(\mathbf{x}^*, \Sigma)$, i.e., without feedback, for pushing. For the same level of disturbances, object's final configurations in the single-finger case are much more diverged from the target. The RMS errors are 0.3727 and 0.0952, respectively.



(a)

(b)

(c)

(d)

Fig. 4. Two composite controllers: (a,c) blending, (b,d) switching.

A. Contact exploitation: Elbow-on-table

The first example considers a robot arm on a table having 4 degrees of freedom. The dynamics of the robot is modeled as a double integrator, where control inputs and disturbances are directly injected as the acceleration of each joint. The goal of this task is to reach a target (red stars in Fig. 2) with its end-effector so the objective function penalizes the squared distance of the end-effector position to the target at the final time as well as the squared control cost along the time horizon ($T = 5$). The problem also involves the inequality constraints to prevent the penetration of robot's body into the table. With the LGP formulation, four different skeletons are considered: reaching the target without touching the table, or while fixing one or two joints on the table for $t \in [3, 5]$, i.e., the inequality constraints for the distance between those joints and the table are activated during that period. Fig. 2 depicts the optimized trajectories with the sample paths from the optimal path distribution. We can observe that, as more degrees of freedom are restricted, the motions are more constrained but the uncertainties are substantially reduced. By constraining its configurations onto the constraint manifold, the robot becomes able to "reject" some disturbances propagating to the end-effector. This is quite realistic, given that tactile sensing from the contact makes it possible to maintain certain contact activities without having the high-gain feedback. Quantitative results are reported in

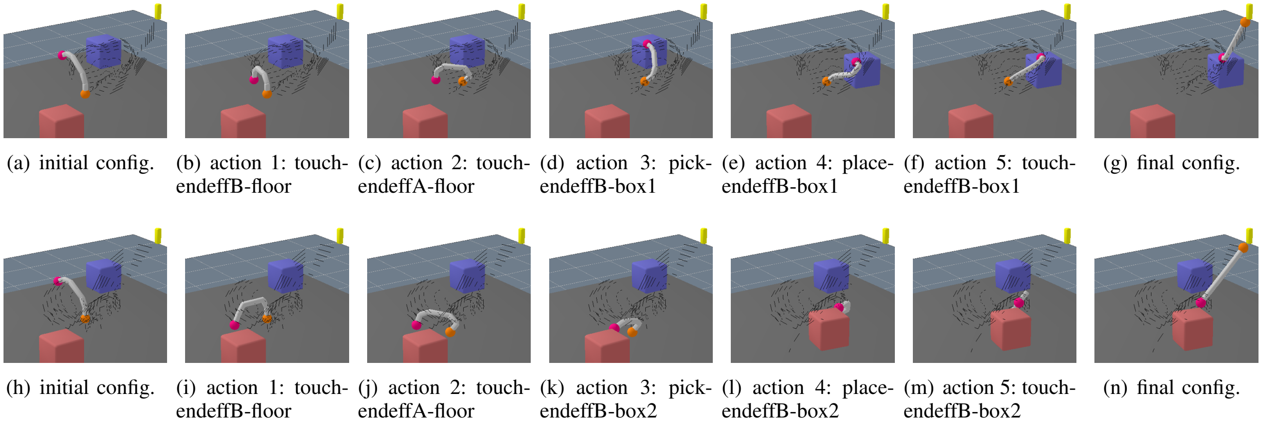

(a) initial config.
(b) action 1: touch-endeffB-floor
(c) action 2: touch-endeffA-floor
(d) action 3: pick-endeffB-box1
(e) action 4: place-endeffB-box1
(f) action 5: touch-endeffB-box1
(g) final config.
(h) initial config.
(i) action 1: touch-endeffB-floor
(j) action 2: touch-endeffA-floor
(k) action 3: pick-endeffB-box2
(l) action 4: place-endeffB-box2
(m) action 5: touch-endeffB-box2
(n) final config.

Fig. 5. Two skeletons for the banana problem. Key frames for (a-g) `(Using Blue Box)` and for (h-i) `(Using Red Box)`.

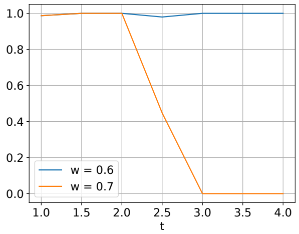


Fig. 6. The mixture weight of `(Using Blue Box)` along time horizon when different disturbances were injected at $t = 2$.

TABLE I
COMPARISONS FOR REACHING, $J^* \approx 2.9180$

	$f(\mathbf{x}^*)$	$\left(\lvert\Sigma_i^*\rvert_+/\lvert\Sigma_i\rvert_+\right)^{1/2}$	$p(\mathbf{a}^*)$
`(Free Reaching)`	**0.1930**	0.0041	0.0626
`(Fixing Joint1)`	0.7682	0.0099	0.0850
`(Fixing Joint2)`	0.6204	0.0584	**0.5810**
`(Fixing Joint1&2)`	1.4827	**0.0646**	0.2713

Table I. As already discussed, `(Free Reaching)` has the lowest deterministic cost, while `(Fixing Joint1&2)` minimizes the stochastic cost. By taking both costs into account as (19), the robot can find the optimal trade-off and choose the best strategy, `(Fixing Joint2)`.

B. Robust planning: Single- vs two-finger push

The second example involves an object to be manipulated, where the goal is to push the object into the target position/orientation using one or two fingers. The robot's dynamics is modeled as a double integrator with 7 degrees of freedom and the motion of the pushed object is defined by the quasi-static dynamics [26], [27], [28]. Under the optimal policy, both plans result in similar deterministic costs, and similarly small RMS errors of the final box configuration, 1.6568×10^{-5} and 1.9720×10^{-5}, respectively. Fig. 3 shows that the two-finger push is inherently more stable, thereby having a lower stochastic cost; the robust strategy can be chosen only when the stochastic cost is also considered.

C. Reactive controller: Elbow-on-table & Banana

For the *Elbow-on-table* example, Fig. 4 shows the executed trajectories with two composite control laws, blending (28) and switching (29). We reduced the control cost weight for KODP to encourage the switching behavior. In both cases, the robot chooses the most constrained strategy, `(Fixing Joint1&2)`, in earlier phases to reduce the uncertainty and shifts to less constrained modes. The blending controller, however, does not make the elbow completely put on the table while the switching does; the robot cannot fully exploit the uncertainty reduction benefit of `(Fixing Joint1&2)` because of this undesirable smoothing effect.

The last example is a so-called banana problem; to catch a banana that is high up, a robot has to move a box first and then climb on it. As depicted in Fig. 5, the robot in the considered scenario can use either the blue or the red box, and `(Using Blue Box)` has a lower cost since it is closer. We injected disturbances in the direction of the red box before the robot takes the first step, and considered the switching composition scheme. Fig. 6 shows that, if the disturbance is small, the robot stays in `(Using Blue Box)`, but switches to `(Using Red Box)` if it is large.

VI. CONCLUSION

This work has proposed a probabilistic framework for manipulation planning in stochastic domains. By connecting hybrid trajectory optimization and approximate posterior inference, we have built the optimal path distribution as a mixture of Gaussians. The proposed framework can evaluate plans not only in the deterministic sense but also in the sense of robustness, allowing for a reactive composite controller.

There is a close connection between this work and LQR-trees [29]. By expanding a backward tree from the goal like LQR-trees, the reactive controller would become able to consider various plans more efficiently. Also, the exponential combinatorial complexity of skeletons (and thus the number of mixture components) can be addressed using deep architectures like [30] while the proposed method also can provide more sensible measures for learning such architectures.

ACKNOWLEDGMENT

We thank the MPI for Intelligent Systems for the Max Planck Fellowship funding.

References

[1] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[2] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[3] D. Mayne, "A second-order gradient method for determining optimal trajectories of non-linear discrete-time systems," *International Journal of Control*, vol. 3, no. 1, pp. 85–95, 1966.

[4] E. Todorov and W. Li, "A generalized iterative lqg method for locally-optimal feedback control of constrained nonlinear stochastic systems," in *Proceedings of the 2005, American Control Conference, 2005.* IEEE, 2005, pp. 300–306.

[5] M. Toussaint, "A tutorial on newton methods for constrained trajectory optimization and relations to slam, gaussian process smoothing, optimal control, and probabilistic inference," in *Geometric and numerical foundations of movements*. Springer, 2017, pp. 361–392.

[6] I. Mordatch, E. Todorov, and Z. Popović, "Discovery of complex behaviors through contact-invariant optimization," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 4, p. 43, 2012.

[7] R. Deits and R. Tedrake, "Footstep planning on uneven terrain with mixed-integer convex optimization," in *2014 IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2014, pp. 279–286.

[8] M. Toussaint, K. Allen, K. A. Smith, and J. B. Tenenbaum, "Differentiable physics and stable modes for tool-use and manipulation planning." in *Robotics: Science and Systems*, 2018.

[9] E. Todorov, "General duality between optimal control and estimation," in *2008 47th IEEE Conference on Decision and Control*. IEEE, 2008, pp. 4286–4292.

[10] K. Rawlik, M. Toussaint, and S. Vijayakumar, "On stochastic optimal control and reinforcement learning by approximate inference," in *Robotics: Science and Systems*, 2012, p. 30523056.

[11] H. J. Kappen, V. Gómez, and M. Opper, "Optimal control as a graphical model inference problem," *Machine learning*, vol. 87, no. 2, pp. 159–182, 2012.

[12] M. Toussaint, "Robot trajectory optimization using approximate inference," in *Proceedings of the 26th annual international conference on machine learning*. ACM, 2009, pp. 1049–1056.

[13] E. Todorov, "Efficient computation of optimal actions," *Proceedings of the national academy of sciences*, vol. 106, no. 28, pp. 11 478–11 483, 2009.

[14] H. J. Kappen and H. C. Ruiz, "Adaptive importance sampling for control and inference," *Journal of Statistical Physics*, vol. 162, no. 5, pp. 1244–1266, 2016.

[15] J.-S. Ha, Y.-J. Park, H.-J. Chae, S.-S. Park, and H.-L. Choi, "Adaptive path-integral autoencoders: Representation learning and planning for dynamical systems," in *Advances in Neural Information Processing Systems*, 2018, pp. 8927–8938.

[16] C. W. Gardiner *et al.*, *Handbook of stochastic methods*. Springer Berlin, 1985, vol. 4.

[17] J.-S. Ha and H.-L. Choi, "A topology-guided path integral approach for stochastic optimal control," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 4605–4612.

[18] T. Erez and E. Todorov, "Trajectory optimization for domains with contacts using inverse dynamics," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 4914–4919.

[19] 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*, vol. 32, no. 9-10, pp. 1164–1193, 2013.

[20] M. Toussaint and M. Lopes, "Multi-bound tree search for logic-geometric programming in cooperative manipulation domains," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 4044–4051.

[21] A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin, *Bayesian data analysis*. Chapman and Hall/CRC, 2013.

[22] A. Levy and R. Rockafellar, "Sensitivity of solutions in nonlinear programming problems with nonunique multipliers," *Recent Advances in Nonsmooth Optimization*, p. 215, 1995.

[23] B. Amos and J. Z. Kolter, "Optnet: Differentiable optimization as a layer in neural networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 136–145.

[24] E. Todorov, "Compositionality of optimal control laws," in *Advances in Neural Information Processing Systems*, 2009, pp. 1856–1864.

[25] U. Muico, J. Popović, and Z. Popović, "Composite control of physically simulated characters," *ACM Transactions on Graphics (TOG)*, vol. 30, no. 3, pp. 1–11, 2011.

[26] M. T. Mason, "Mechanics and planning of manipulator pushing operations," *The International Journal of Robotics Research*, vol. 5, no. 3, pp. 53–71, 1986.

[27] J. Zhou, M. T. Mason, R. Paolini, and D. Bagnell, "A convex polynomial model for planar sliding mechanics: theory, application, and experimental validation," *The International Journal of Robotics Research*, vol. 37, no. 2-3, pp. 249–265, 2018.

[28] M. Toussaint, J.-S. Ha, and D. Driess, "Describing physics for physical reasoning: Force-based sequential manipulation planning," *arXiv preprint arXiv:2002.12780*, 2020.

[29] R. Tedrake, I. R. Manchester, M. Tobenkin, and J. W. Roberts, "LQR-trees: Feedback motion planning via sums-of-squares verification," *The International Journal of Robotics Research*, vol. 29, no. 8, pp. 1038–1052, 2010.

[30] D. Driess, O. Oguz, J.-S. Ha, and M. Toussaint, "Deep visual heuristics: Learning feasibility of mixed-integer programs for manipulation planning," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.