Data-Centric Engineering (2025), 6: e41
doi:10.1017/dce.2025.10021

RESEARCH ARTICLE

Quasi-static contact-rich manipulation using dynamic movement primitives and haptic potential map

Huu-Thiet Nguyen, Lin Yang, Chen Lv and Domenico Campolo

School of Mechanical and Aerospace Engineering, Nanyang Technological University, Singapore, Singapore
Corresponding author: Domenico Campolo; Email: d.campolo@ntu.edu.sg

Received: 05 October 2024; **Revised:** 13 July 2025; **Accepted:** 13 July 2025

Keywords: contact-rich manipulation; Gaussian process regression; haptic DMPs; physics-informed

Abstract

Robots need a sense of touch to handle objects effectively, and force sensors provide a straightforward way to measure touch or physical contact. However, contact force data are typically sparse and difficult to analyze, as it only appears during contact and is often affected by noise. Therefore, many researchers have consequently relied on vision-based methods for robotic manipulation. However, vision has limitations, such as occlusions that block the camera's view, making it ineffective or insufficient for dexterous tasks involving contact. This article presents a method for robotic systems operating under quasi-static conditions to perform contact-rich manipulation using only force/torque measurements. First, the interaction forces/torques between the manipulated object and its environment are collected in advance. A potential function is then constructed from the collected force/torque data using Gaussian process regression with derivatives. Next, we develop haptic dynamic movement primitives (Haptic DMPs) to generate robot trajectories. Unlike conventional DMPs, which primarily focus on kinematic aspects, our Haptic DMPs incorporate force-based interactions by integrating the constructed potential energy. The effectiveness of the proposed method is demonstrated through numerical tasks, including the classical peg-in-hole problem.

Impact Statement

Visual simultaneous localization and mapping (SLAM) has been intensively studied and widely adopted over the years, becoming a key component of mobile robot navigation. However, when it comes to robotic manipulation —where navigation involves moving the arm and end-effector through contact-rich environments to accomplish tasks—vision-only approaches can easily fall short. Humans, on the other hand, can perform dexterous manipulation tasks using only haptic sensing. Inspired by the environment mapping technique in visual SLAM, this article proposes to establish a potential energy distribution via premeasured contact forces and torques. Using this distribution, the robot's trajectory can be generated, helping it navigate the contact-rich environment and accomplish the task. The approach has the potential to change the way contact-rich tasks are executed in the future.

1. Introduction

In robotics, physical contact (or simply contact) occurs when a part of the robot or the object it handles touches other objects, humans, or the environment. Depending on the application, this contact can be

H.-T.N. and L.Y. these authors contributed equally to this work.

accidental or intentional. Avoiding unwanted contact is crucial in applications such as collision avoidance (Koptev et al., 2021), whereas in other cases, contact is either inevitable or deliberately introduced (Suomalainen et al., 2022). In robotic assembly, maintaining contact with objects is essential for executing tasks such as insertion and drilling (Suomalainen et al., 2022; Yang et al., 2025; Yang et al., 2025). Robotic maneuvring that involves contact requires careful trajectory planning to ensure successful task execution. However, traditional planning methodologies often focus on avoiding direct contact, aiming for collision-free navigation (LaValle, 2006). In contact-rich manipulation, while preventing accidental collisions is crucial, occasional or deliberate contact is essential for task completion. Proper contact modeling can enhance contact-rich manipulation planning, but modeling contact between objects remains challenging due to its complex and nonlinear nature (Wirnshofer et al., 2019). Therefore, it is important to establish a systematic framework for modeling and planning in contact-rich manipulation, enabling robots to effectively navigate contact-rich environments during manipulation tasks.

Vision-based approaches for robotic mapping, planning (Garg et al., 2020), and manipulation (Ehsani et al., 2021) have attracted much attention from roboticists, aligning with the rapid development of computer vision applications in the last decade. The use of cameras has also led to the emergence and growth of visual simultaneous localization and mapping (Kazerouni et al., 2022; Kok et al., 2024). However, vision systems are prone to failure in many scenarios, such as poor lighting conditions or visual obstructions. These drawbacks are more apparent during contact, as the geometries or shapes of objects can create shadows on themselves and affect the cameras' field of view. In such cases, other sensing techniques, such as haptics, may help. Humans also rely on haptics, or the sense of touch, to handle objects, especially when visibility is limited.

For contact-rich robotic manipulation, haptic information plays an essential role in perception, planning, and control (Diana and Marescaux, 2015; Dawson-Elli and Adamczyk, 2020). Various devices provide haptic feedback, including tactile sensors and force/torque sensors. Tactile sensors are designed to mimic the sense of touch in human skin, often incorporating specialized materials and techniques (Girão et al., 2013). They are mostly used for in-hand robotic perception, allowing robots to recognize objects through touch, similar to how humans explore their environment. For example, Bhattacharjee et al. (n.d.) combined tactile sensing with vision to label pixels on visible surfaces using hidden Markov models. Another study (Strub et al., n.d.) used tactile data from a two-fingered robotic hand manipulating a polygon to infer the object's shape. On the other hand, force/torque sensors measure the magnitude of contact forces and torque. Unlike tactile sensors, which typically generate high-dimensional outputs, force/torque sensors provide much lower-dimensional data (typically six-dimensional) and are commonly integrated into robot arms by manufacturers. Force measurements are particularly important in force-based control strategies, such as impedance control (Caccavale et al., 1999), and in manipulation tasks that require precise force regulation, such as robotic assembly of delicate objects. However, force feedback is typically sparse as it is only available upon contact. This contrasts with vision-based data, where cameras can continuously capture a scene from a wide range of distances. Moreover, force data are often noisy due to factors, such as torque ripple, mechanical vibrations, and electrical interference (Katsura et al., 2007; Turlapati et al., 2024). Therefore, constructing an environment map solely from force feedback is a challenging task. Some recent studies have explored the use of force/torque sensors to infer geometric information. For instance, Turlapati and Campolo (2022) estimate object kinematics by integrating haptic feedback at contact points with an initial vision-based pose estimate. Another work (Suresh et al., 2021) employs Gaussian process (GP) implicit surfaces on contact data to infer object geometry from planar pushing. While these approaches demonstrate the potential of haptics for geometric reasoning, a fully realized environment map derived purely from force/torque sensors, particularly for general manipulation tasks involving motion planning, has yet to be developed.

When a map is created that captures all objects in the robot's operating space with which the robot can interact, trajectory planning can be performed for a manipulation task. Dynamic movement primitives (DMPs) (Ijspeert et al., 2013) have been widely adopted as an effective method for robot trajectory generation. DMPs model complex movements as nonlinear dynamical systems, incorporating stable behavior while allowing flexibility to follow learnt trajectories. DMPs have demonstrated robustness and

generalizability properties, especially in the context of learning from demonstration (LfD) (Saveriano et al., 2023). By adding a coupling term, DMPs can also be adapted for learning the feedback force during interactions. For instance, Gams et al. (2014) showed that this coupling term allows DMPs to handle interaction forces from the environment or another robot arm. However, their work did not address more complex tasks, such as assembly, which may involve multiple contact points or intermittent contact. It has also been shown that assembly tasks can be accomplished through DMPs and LfD (Wang et al., 2022; Zhao et al., 2023). While LfD is an effective method for quick imitation of a task, the underlying policy, intention, or skill is difficult to extract from LfD, making it challenging to generalize to a broader class of similar tasks. This challenge arises because LfD primarily replicates movements from kinematic demonstrations, rather than teaching a systematic strategy for why one trajectory is chosen over another. Another approach for learning the parameters of the DMPs is through black-box optimization (BBO) (Stulp and Sigaud, 2013). A recent study demonstrated that a peg-in-hole task can be accomplished by combining DMPs with BBO algorithms (Yang et al., 2023, 2025).

In modeling manipulation tasks, a commonly used method is the quasi-static assumption, where the system is assumed to be in equilibrium and inertial effects are minimal (Whitney et al., 1982; Katayarna et al., 2022). Under this assumption, Campolo and Cardin (Campolo and Cardin, n.d., 2025) introduced a potential-based framework for quasi-static manipulation that conceptualizes it as a planning problem on an implicit manifold. In this framework, the configuration space is split into two types of variables: directly controllable variables (or control inputs) and indirectly controllable variables (or internal states). The core concept revolves around the system's total potential energy, represented as a function of these variables. Under quasi-static conditions, the planning problem is reduced to finding trajectories on an equilibrium manifold (EM), a set of points where the gradient of the potential energy with respect to the internal states equals zero. Using this framework, contact-rich manipulation can be smoothly analyzed in the space of directly controllable variables. The internal states then adjust to follow these variables along the EM. This approach differs from traditional methods that rely on predefined discrete stages or contact modes, such as the four-stage framework (approaching, searching, aligning, and inserting) with a different number of contact points (e.g., no contact, two-point contact, and three-point contact) (Lee et al., 2022).

This study employs the framework where the potential energy is assumed to comprise two parts: (i) internal control energy, which results from the displacement between the desired (directly controllable) and actual (indirectly controllable) positions of the object, and (ii) external interaction energy, which arises from interactions such as elastic contact with the environment or gravitational forces. The external potential is built from premeasured forces/torques via GP regression (GPR) with derivatives (Solak et al., 2002). Haptic-DMPs are proposed to generate manipulation trajectories based on the constructed potential function, guiding the robot to accomplish the manipulation tasks. In summary, the contributions of this study are as follows:

- The development of a haptic potential map of the environment based on observations of contact forces at various positions in space, using GPR with derivatives. The use of GPR to derive energy from forces makes the model resemble a physics-based machine.
- The development of haptic DMPs (Haptic-DMPs that account for elastic interactions between the robot and the object, as well as the object and the environment. Haptic-DMPs solve ordinary differential equations (ODEs) to determine the object's path from the control trajectory.
- A systematic framework for trajectory generation in contact-rich manipulation tasks through the combination of GPR, Haptic-DMPs, and BBO.

2. Quasi-static mechanical manipulation

Consider a mechanical system governed solely by conservative forces (an example of such a system is shown in Figure 1). Adopting the framework proposed by Campolo and Cardin (n.d., 2025), where the configuration space can be divided into directly controllable and indirectly controllable variables, with

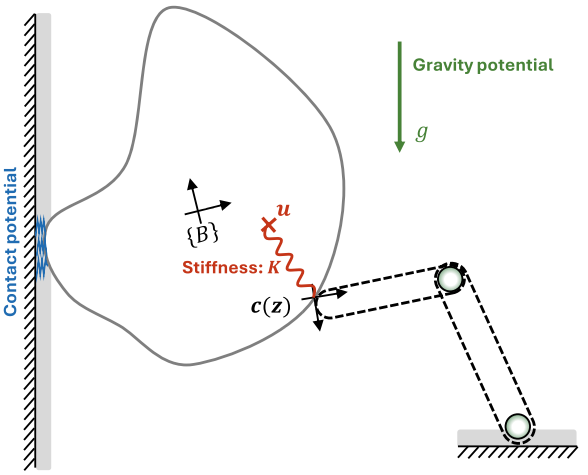


Figure 1. *A robot manipulating an object while trying to maintain the object's contact with the wall. Our framework considers both the indirectly controllable variable (state $\mathbf{z}$) and the directly controllable variable (control input $\mathbf{u}$). In this specific problem, $\mathbf{u} = \begin{bmatrix} u_x, u_y \end{bmatrix}^T$ is the desired position of the robot's end effector, $\mathbf{z} = \begin{bmatrix} z_x, z_y, z_\theta \end{bmatrix}^T$ is the pose of the object (its xy-coordinates and θ-rotation), and $\mathbf{c}(\mathbf{z})$ is the function that converts the pose of the object to the location of the contact point with the robot's end effector.*

$\boldsymbol{z} \in \mathcal{Z} \subset \mathbb{R}^N$ denoting the *internal states* (indirectly controllable) and $\boldsymbol{u} \in \mathcal{U} \subset \mathbb{R}^K$ being the *control inputs* (directly controllable), there exists a total potential energy of the system as

$$W(\boldsymbol{z}, \boldsymbol{u}) : \mathcal{Z} \times \mathcal{U} \rightarrow \mathbb{R} - 2mm \tag{2.1}$$

This total potential can be partitioned into $W_{ctrl}(\boldsymbol{z}, \boldsymbol{u})$—the elastic energy arising from the displacement between the control inputs $\boldsymbol{u}$ and the object's states $\boldsymbol{z}$, and $U(\boldsymbol{z})$—the energy resulting from the interaction between the object and its environment.

$$W(\boldsymbol{z}, \boldsymbol{u}) = W_{ctrl}(\boldsymbol{z}, \boldsymbol{u}) + U(\boldsymbol{z}) \tag{2.2}$$

The term $W_{ctrl}(\boldsymbol{z}, \boldsymbol{u})$ in Equation (2.2) can be described as

$$W_{ctrl}(\boldsymbol{z}, \boldsymbol{u}) = \frac{1}{2} \|\boldsymbol{u} - \boldsymbol{c}(\boldsymbol{z})\|_{\boldsymbol{K}}^2 \triangleq \frac{1}{2} (\boldsymbol{u} - \boldsymbol{c}(\boldsymbol{z}))^T \boldsymbol{K} (\boldsymbol{u} - \boldsymbol{c}(\boldsymbol{z})) \tag{2.3}$$

where $\boldsymbol{K}$ is the stiffness matrix of the virtual spring and $\boldsymbol{c}(\boldsymbol{z}) : \mathcal{Z} \rightarrow \mathcal{U}$ is a conversion function.

The term $U(\boldsymbol{z})$ includes components such as gravitational potential U_g and elastic contact potential U_c. This study focuses on contact interactions, so we consider scenarios where U_g hardly changes during tasks and can thus be treated as negligible. For example, in tasks where manipulated objects are restricted to a horizontal surface, the vertical position (z_z), which represents the object's position along the vertical axis, does not change. This allows U_g, typically expressed as $mgz_z + \zeta$ (where m is the mass of the object, g is the gravitational acceleration, and ζ is a constant offset), to be set to zero by choosing an appropriate constant (ζ). In contrast, the contact potential, U_c, depends on various factors, such as the materials of the object and its environment, and their shapes or geometries, making it very difficult to compute $U_c(\boldsymbol{z})$ explicitly. Therefore, we rely on GPR to approximate $U_c(\boldsymbol{z})$, which is presented in the next section. For simplicity, throughout the rest of the article, U refers to U_c.

2.1. Equilibrium manifold

Once the potential function $W(\boldsymbol{z}, \boldsymbol{u})$ is explicitly defined, we can consider quasi-static manipulation as maneuvring on the so-called EM, which contains all mechanical equilibria $\boldsymbol{z}^*$ as the solutions of $\nabla_{\boldsymbol{z}} W(\boldsymbol{z}, \boldsymbol{u}) = \boldsymbol{0}$. In other words,

$$\nabla_z W\left(\boldsymbol{z}_m^*, \boldsymbol{u}\right) = \mathbf{0} \in \mathbb{R}^N. \tag{2.4}$$

We define $\nabla_q W \equiv \left[\partial_{q_1} W, \ldots, \partial_{q_a} W\right]^T$, where the *nabla* (column) operator is defined as $\nabla_q = \left[\partial_{q_1}, \ldots, \partial_{q_a}\right]^T$. Meanwhile, we define the shorthand notation $\nabla_z^2 \equiv \left(\nabla_z \nabla_z^T\right) = \nabla_z \nabla_z^T$ for Hessians and mixed-derivative operators. Here, the subscript m represents possibly different solutions for a single value of $\boldsymbol{u}$. This means that for the same control input, the object can have different positions or states in equilibrium. As such, the EM is defined when $\boldsymbol{u}$ is seen as a parameter, that is,

$$\mathcal{M}^{eq} \triangleq \left\{(\boldsymbol{z}, \boldsymbol{u}) \in \mathcal{Z} \times \mathcal{U} \,|\, \nabla_z W(\boldsymbol{z}, \boldsymbol{u}) = \mathbf{0}\right\} \tag{2.5}$$

is a smooth embedded submanifold in the ambient space $\mathcal{Z} \times \mathcal{U}$. The state transitions are controlled purely by the robotic agent $\boldsymbol{u}$. Meanwhile, a point is strictly stable when its Hessian is positive definite, that is, $\nabla_{zz}^2 W\big|_* \succ 0$.

3. GPR with derivatives for mapping the environment

In this section, we map the environment by means of potential energy. More specifically, we aim to approximate $U(\boldsymbol{z})$ in Equation (2.2) from observations of contact forces/torques. Now, let us consider a general case, an unknown but differentiable scalar function $f : \mathbb{R}^N \to \mathbb{R}$; $\boldsymbol{z} \mapsto f(\boldsymbol{z})$ for which readings are available at specific locations. Normal regression can be used to estimate the function $y = f(\boldsymbol{z})$ by using its observed values. However, in some scenarios, the observations of y can be difficult to obtain or even infeasible, but the observations of derivatives $\partial y / \partial z_i$ are abundant. The contact energy $U(\boldsymbol{z})$ in Equation (2.2) is an example where direct measurement is infeasible, but its derivatives, forces, and torques collectively denoted as $\boldsymbol{F}$, are measurable, noting that $-\nabla U(\boldsymbol{z}) = \boldsymbol{F}$ in conservative systems. In this case, GPR can be used. We will look at how this can be done, from reviewing normal GPR using function value measurements to adapting to deal with function derivative measurements.

A GP is normally described as

$$f(\boldsymbol{z}) \sim \mathcal{GP}(\mu(\boldsymbol{z}), \kappa(\boldsymbol{z}, \boldsymbol{z}')) \tag{3.1}$$

where $\mu : \mathbb{R}^N \to \mathbb{R}$ is the mean function, and $\kappa : \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}$ is the covariance function (or kernel) $\kappa(\boldsymbol{z}, \boldsymbol{z}') = \mathrm{cov}(f(\boldsymbol{z}), f(\boldsymbol{z}'))$. Additionally, for any finite set of input points, the corresponding function values follow a multivariate normal (Gaussian) distribution (Williams and Rasmussen, 2006). Using the concept of GP, the GPR can be formulated as follows:

Given S noisy observations $y_1, y_2, \ldots, y_S$ of $f(\boldsymbol{z})$ at corresponding points $\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_S$, we want to estimate the unknown function $f(\boldsymbol{z})$ by regression. In this case, we can consider that $f(\boldsymbol{z})$ follows a GP with zero mean, that is, $f(\boldsymbol{z}) \sim \mathcal{GP}(0, \kappa(\boldsymbol{z}, \boldsymbol{z}'))$. Denoting $\boldsymbol{y} = [y_1 \; y_2 \; \ldots \; y_S]^T \in \mathbb{R}^{S \times 1}$, the predicted value y^* at a new input $\boldsymbol{z}^*$ are then given by its mean $\overline{y^*}$ and variance $\mathbb{V}[y^*]$, or $y^* \sim \mathcal{N}(\overline{y^*}, \mathbb{V}[y^*])$, calculated as

$$\overline{y^*} = \boldsymbol{k}^{*T} \left(\boldsymbol{K} + \sigma_n^2 \boldsymbol{I}\right)^{-1} \boldsymbol{y} \tag{3.2}$$

$$\mathbb{V}[y^*] = k^{**} - \boldsymbol{k}^{*T} \left(\boldsymbol{K} + \sigma_n^2 \boldsymbol{I}\right)^{-1} \boldsymbol{k}^* \tag{3.3}$$

Here, $\boldsymbol{K} = [\kappa_{ii'}] \in \mathbb{R}^{S \times S}$ with $\kappa_{ii'} = \kappa(\boldsymbol{z}_i, \boldsymbol{z}_{i'}), 1 \le i, i' \le S$ is the covariance matrix; $\boldsymbol{k}^* = [\kappa_i^*] \in \mathbb{R}^{S \times 1}$ with $\kappa_i^* = \kappa(\boldsymbol{z}^*, \boldsymbol{z}_i) \in \mathbb{R}, 1 \le i \le S$; $k^{**} = \kappa(\boldsymbol{z}^*, \boldsymbol{z}^*) \in \mathbb{R}$; σ_n^2 is the variance of the noise in the observations; and $\boldsymbol{I} \in \mathbb{R}^{S \times S}$ is the identity matrix.

Remark 1: For simplicity, this study assumes a zero mean function a priori. This assumption is particularly suitable when no prior knowledge of the function to be approximated is available. However, if f follows Equation (3.1) with a nonzero prior mean function μ, a change of variable to $f' = f - \mu$ ensures that f' follows $\mathcal{GP}(0, \kappa)$.

3.1. GPR with derivatives

An important and favorable property of GPs is their closure under linear operations (De Roos et al., 2021). As differentiation is a linear operator, the derivative of a GP is also a GP (Solak et al., 2002). Taking into account the derivatives of $f(\boldsymbol{z})$, the covariance functions in Equation (3.1) should follow the equations below for each element z_i of vector $\boldsymbol{z}$ and elements z_i' and z_j' of vector $\boldsymbol{z}'$

$$\operatorname{cov}\left(\frac{\partial f(\boldsymbol{z})}{\partial z_i}, f(\boldsymbol{z}')\right) = \frac{\partial \kappa(\boldsymbol{z},\boldsymbol{z}')}{\partial z_i},\ \operatorname{cov}\left(\frac{\partial f(\boldsymbol{z})}{\partial z_i}, \frac{\partial f(\boldsymbol{z}')}{\partial z_j'}\right) = \frac{\partial^2 \kappa(\boldsymbol{z},\boldsymbol{z}')}{\partial z_i \partial z_j'} \tag{3.4}$$

Applying for the vector $\boldsymbol{z}$ results in the gradient vector ∇ and the Hessian matrix ∇^2. Now, we can rewrite Equation (3.1) as

$$\begin{bmatrix} f(\boldsymbol{z}) \\ \nabla f(\boldsymbol{z}) \end{bmatrix} \sim \mathcal{GP}\left(\begin{bmatrix} \mu(\boldsymbol{z}) \\ \nabla \mu(\boldsymbol{z}) \end{bmatrix}, \begin{bmatrix} \kappa(\boldsymbol{z},\boldsymbol{z}') & \nabla_{\boldsymbol{z}'}^T \kappa(\boldsymbol{z},\boldsymbol{z}') \\ \nabla_{\boldsymbol{z}} \kappa(\boldsymbol{z},\boldsymbol{z}') & \nabla^2 \kappa(\boldsymbol{z},\boldsymbol{z}') \end{bmatrix}\right) \tag{3.5}$$

With this in mind, now assume that we have S scalar (noisy) readings $\{y_1,\ldots,y_S\}$ at S locations $\{\boldsymbol{z}_1,\ldots,\boldsymbol{z}_S\}$, and M (noisy) gradients $\left\{\nabla y_1^\diamond,\ldots,\nabla y_M^\diamond\right\}$ at M locations $\left\{\boldsymbol{z}_1^\diamond,\ldots,\boldsymbol{z}_M^\diamond\right\}$. The extended observation vector is now $\breve{\boldsymbol{y}} \triangleq [y_1 \ \ y_2 \ \ \ldots \ \ y_S \ \ \nabla y_1^\diamond \ \ \ldots \ \ \nabla y_M^\diamond]^T \in \mathbb{R}^{(S+MN)\times 1}$. The covariance matrix is now built as follows

$$\breve{\boldsymbol{K}} = \begin{bmatrix} \boldsymbol{K}_{11} & \boldsymbol{K}_{12} \\ \boldsymbol{K}_{21} & \boldsymbol{K}_{22} \end{bmatrix} \in \mathbb{R}^{(S+MN)\times(S+MN)}, \text{where} \tag{3.6}$$

$$\begin{aligned}
&\boldsymbol{K}_{11} = [\kappa_{ii'}] \in \mathbb{R}^{S\times S} \text{ with } \kappa_{ii'} = \kappa(\boldsymbol{z}_i,\boldsymbol{z}_{i'}) \in \mathbb{R}, 1 \le i,i' \le S, \\
&\boldsymbol{K}_{12} = [\boldsymbol{\kappa}_{ij}] \in \mathbb{R}^{S\times MN} \text{ with } \boldsymbol{\kappa}_{ij} = \nabla_{\boldsymbol{z}_j^\diamond}^T \kappa\left(\boldsymbol{z}_i,\boldsymbol{z}_j^\diamond\right) \in \mathbb{R}^{1\times N}, 1 \le i \le S, 1 \le j \le M, \\
&\boldsymbol{K}_{21} = [\boldsymbol{\kappa}_{ji}] \in \mathbb{R}^{MN\times S} \text{ with } \boldsymbol{\kappa}_{ji} = \nabla_{\boldsymbol{z}_i} \kappa\left(\boldsymbol{z}_j^\diamond,\boldsymbol{z}_i\right) \in \mathbb{R}^{N\times 1}, 1 \le j \le M, 1 \le i \le S, \\
&\boldsymbol{K}_{22} = \left[\boldsymbol{\kappa}_{jj'}\right] \in \mathbb{R}^{MN\times MN} \text{ with } \boldsymbol{\kappa}_{jj'} = \nabla^2 \kappa\left(\boldsymbol{z}_j^\diamond,\boldsymbol{z}_{j'}^\diamond\right) \in \mathbb{R}^{N\times N}, 1 \le j,j' \le M.
\end{aligned}$$

Assuming that the prior unknown function $y = f(\boldsymbol{z})$ follows a GP with zero mean, that is, $\mu(\boldsymbol{z}) = 0$ and $\nabla\mu(\boldsymbol{z}) = \boldsymbol{0}$, similar to Equation (3.2), the mean of the posterior y^* can be calculated as

$$\overline{y^*} = \breve{\boldsymbol{k}}^{*T}\left(\breve{\boldsymbol{K}} + \sigma_n^2 \boldsymbol{I}\right)^{-1} \breve{\boldsymbol{y}}, \text{with} \tag{3.7}$$

$$\breve{\boldsymbol{k}}^{*T} = \left[\boldsymbol{k}_1^{*T} \ \ \boldsymbol{k}_2^{*T}\right] \in \mathbb{R}^{1\times(S+MN)} \tag{3.8}$$

where $\boldsymbol{k}_1^{*T} = [\kappa_i^*] \in \mathbb{R}^{1\times S}$ with $\kappa_i^* = \kappa(\boldsymbol{z}_i,\boldsymbol{z}^*) \in \mathbb{R}, 1 \le i \le S$, and

$\boldsymbol{k}_2^{*T} = \left[\boldsymbol{\kappa}_j^*\right] \in \mathbb{R}^{1\times MN}$ with $\boldsymbol{\kappa}_j^* = \nabla_{\boldsymbol{z}_j^\diamond}^T \kappa\left(\boldsymbol{z}_j^\diamond,\boldsymbol{z}^*\right) \in \mathbb{R}^{1\times N}, 1 \le j \le M$.

Given a fixed set of observations, it should follow that $\breve{\boldsymbol{\alpha}} \triangleq \left(\breve{\boldsymbol{K}} + \sigma_n^2 \boldsymbol{I}\right)^{-1} \breve{\boldsymbol{y}} \in \mathbb{R}^{(S+MN)\times 1}$ is a constant vector. We can rewrite Equation (3.7) as

$$\overline{y^*} = \breve{\boldsymbol{k}}^{*T} \times \breve{\boldsymbol{\alpha}} \tag{3.9}$$

Taking derivatives with respect to $\boldsymbol{z}^*$ gives $\overline{\nabla_{\boldsymbol{z}^*} y^*} = \nabla_{\boldsymbol{z}^*} \breve{\boldsymbol{k}}^{*T} \times \breve{\boldsymbol{\alpha}}$ and $\overline{\nabla_{\boldsymbol{z}^*}^2 y^*} = \nabla_{\boldsymbol{z}^*}^2 \breve{\boldsymbol{k}}^{*T} \otimes \breve{\boldsymbol{\alpha}}$. Simplifying the notations

$$\overline{\nabla y^*} = \nabla \breve{\boldsymbol{k}}^{*T} \times \breve{\boldsymbol{\alpha}} \in \mathbb{R}^{N\times 1} \tag{3.10}$$

$$\overline{\nabla^2 y^*} = \nabla^2 \breve{\boldsymbol{k}}^{*T} \otimes \breve{\boldsymbol{\alpha}} \in \mathbb{R}^{N\times N} \tag{3.11}$$

where $\nabla \breve{\boldsymbol{k}}^{*T} \in \mathbb{R}^{N \times (S+MN)}$, and $\nabla^2 \breve{\boldsymbol{k}}^{*T} \in \mathbb{R}^{N \times N \times (S+MN)}$ (Dattorro, 2010). Here, $\otimes$ denotes tensor multiplication, which contracts the third dimension of $\nabla^2 \breve{\boldsymbol{k}}^{*T}$ with the first dimension of $\breve{\boldsymbol{\alpha}}$.

3.2. Approximation of U(z) using GPR with derivatives

In this work, the force/torque observations are obtained by maneuvring the object within the environment and recording the corresponding positions and forces/torques. This data collection involves instances where the object is either in free space, that is, no contact, or in contact with the environment. These force and torque observations represent derivatives of $U(\boldsymbol{z})$, as $-\nabla U(\boldsymbol{z}) = \boldsymbol{F}$. The energy function $U(\boldsymbol{z})$ is then constructed using GPR. The incorporation of derivatives makes the GPR resemble a physics-based model (Cross et al., 2024). Using Equation (3.5), $U(\boldsymbol{z})$ is modeled as a GP with zero mean,

$$\begin{bmatrix} U(\boldsymbol{z}) \\ \nabla U(\boldsymbol{z}) \end{bmatrix} \sim \mathcal{GP}\left(\begin{bmatrix} 0 \\ \boldsymbol{0} \end{bmatrix}, \begin{bmatrix} \kappa(\boldsymbol{z},\boldsymbol{z}') & \nabla_{\boldsymbol{z}'}^T \kappa(\boldsymbol{z},\boldsymbol{z}') \\ \nabla_{\boldsymbol{z}} \kappa(\boldsymbol{z},\boldsymbol{z}') & \nabla^2 \kappa(\boldsymbol{z},\boldsymbol{z}') \end{bmatrix} \right) \tag{3.12}$$

Using Equations (3.9), (3.10), and (3.11), we can estimate the function $U(\boldsymbol{z})$ and its first- and second-order gradients $\nabla U(\boldsymbol{z})$, $\nabla^2 U(\boldsymbol{z})$ respectively.

Remark 2: The current study represents an initial exploration in which we primarily utilize the posterior predictive mean of GPR to map the environment and derive a control policy for robot manipulation tasks. Incorporating the posterior predictive covariance within this framework can enhance uncertainty-aware decision-making in these tasks and remains an open area for further investigation.

Remark 3: A key limitation of conventional GPR is its computational complexity, which scales poorly with the number of observations and may become a bottleneck in large-scale scenarios. The current framework focuses on offline training of the GPR model using pre-collected data, followed by an offline policy search based on the posterior predictive mean using Haptic-DMPs (as depicted in Figure 2). Policy execution, however, can be performed in real time. Reducing computational complexity remains an important consideration, and approaches such as scalable GPR (Liu et al., 2020) offer potential solutions.

4. Haptic-DMPs

With the contact potential $U(\boldsymbol{z})$ serving as an environment map and now approximated using GPR on contact force/torque observations (left part of Figure 2), we are ready to plan the robot's motion. In this section, we introduce Haptic-DMPs and the BBO technique used to train their parameters (right part of Figure 2). The key idea is to plan within the space of directly controllable variables while ensuring that the entire system (both $\boldsymbol{u}$ and $\boldsymbol{z}$) remains on the implicitly defined EM, following an approach similar to our

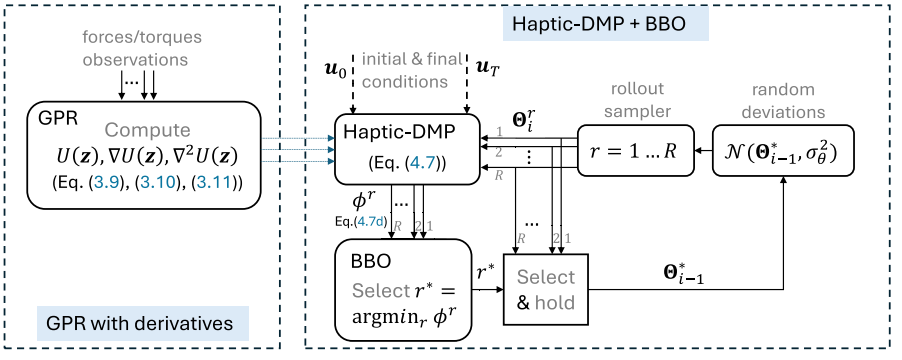


Figure 2. *Planning framework with GPR, Haptic-DMPs, and BBO. BBO selects the optimal parameter* Θ *from R random deviations from the optimal parameter* Θ_{i-1}^* *of the previous iteration.*

previous work on continuation (Yang et al., 2025). Before introducing Haptic-DMPs, we briefly revisit the standard DMP formulation.

4.1. Dynamic movement primitives

DMPs have been introduced (Ijspeert et al., 2013) as an attractor for a simple second-order linear system, such as a damped spring model $\tau\ddot{y} = \alpha_z\left(\beta_z(g-y) - \dot{y}\right) + f$ with y is the position and g is the goal position. With z being the velocity, the dynamic equation can be rewritten in first-order notation as

$$\begin{cases} \tau\dot{z} = \alpha_z\left(\beta_z(g-y) - z\right) + f \\ \tau\dot{y} = z \end{cases} \tag{4.1}$$

where $\tau > 0$ is a time scaling constant, α_z and β_z are positive constants, and f is a forcing term. f can be chosen as a linear combination of P nonlinear radial basis functions:

$$f(x) = \frac{\sum_{i=1}^{P} \Psi_i(x)\theta_i}{\sum_{i=1}^{P} \Psi_i(x)} x \tag{4.2}$$

with θ_i being adjustable weights (or parameters), and the basis functions as $\Psi_i(x) = \exp\left(-h_i(x-c_i)^2\right)$, where h_i and c_i are constants that determine the width and centers of the basis functions, and x is a phase variable, as the solution of the canonical system $\tau\dot{x} = -\alpha_x x$.

The weights $\boldsymbol{\Theta} = [\theta_1, \ldots, \theta_P]^T$ of the DMP can be learnt or calculated such that the resulted trajectory meets the task requirements.

4.2. Haptic-DMPs for quasi-static systems

For a quasi-static system with the total energy described by Equation (2.2), we will perform motion planning for directly controllable variables $\boldsymbol{u}$. This approach is necessary because the EM is implicitly defined, meaning that the state variables $\boldsymbol{z}(\boldsymbol{u})$ are not explicitly known. In real-world scenarios, this reflects the fact that the exact state $\boldsymbol{z}$ of objects cannot be determined before issuing the control command $\boldsymbol{u}$. Therefore, a method is required to explore and compute $\boldsymbol{z}(t)$.

Given any control state $\boldsymbol{u}_0 \in \mathbb{R}^K$, our goal is to compute the equilibrium state $\boldsymbol{z}^*$ satisfying the quasi-static condition $\nabla_z W\left(\boldsymbol{z}_0^*, \boldsymbol{u}_0\right) = \boldsymbol{0}$. Starting from an initial guess $\boldsymbol{z}_0$, $\boldsymbol{z}^*$ can be computed by an ODE using Newton's method (or natural gradient) as below:

$$\begin{cases} \dfrac{d}{dt}\boldsymbol{z}(t) = -\eta\left(\nabla_{zz}^2 W\right)^{-1}\nabla_z W \\ \boldsymbol{z}(0) = \boldsymbol{z}_0 \end{cases} \tag{4.3}$$

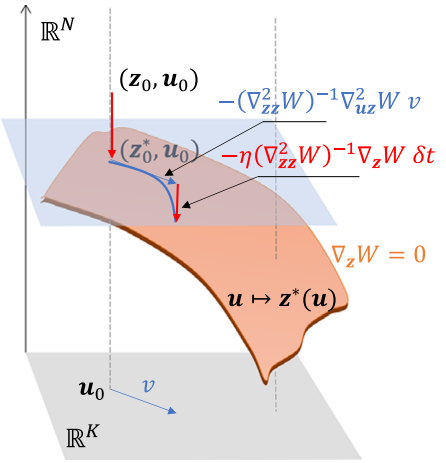
where η is a positive scalar representing step size in the ODE (Schneebeli and Wihler, 2011), influencing the convergence of the ODE. If η is too small, the ODE may not converge in a given time. Conversely, if η is too large, the ODE could jump to a different branch of the EM. This ODE is presented by the red vector from $(\boldsymbol{z}_0, \boldsymbol{u}_0)$ to $\left(\boldsymbol{z}_0^*, \boldsymbol{u}_0\right)$ in Figure 3(a).

Since all valid states must remain on this manifold, finding a control policy requires exploration within it. However, we can only directly control $\boldsymbol{u}$, while $\boldsymbol{z}(\boldsymbol{u})$ remains implicit. Therefore, we need an approach that enables evolution on the EM. For nonconstant $\boldsymbol{u}(t)$, from the equilibrium condition $\nabla_z W(\boldsymbol{z}, \boldsymbol{u}) = \boldsymbol{0}$, differentiating with respect to time t, we have

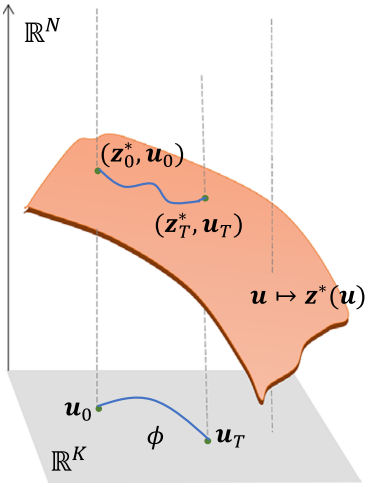
$$\nabla_{zz}^2 W \delta\boldsymbol{z} + \nabla_{uz}^2 W \delta\boldsymbol{u} = 0 \tag{4.4}$$

$$\Rightarrow \delta\boldsymbol{z} = -\left(\nabla_{zz}^2 W\right)^{-1}\nabla_{uz}^2 W \delta\boldsymbol{u} \tag{4.5}$$

which describes the tangent variation. This behavior is represented by the blue vector in Figure 3(a), which lies in the tangent space at the point $\left(\boldsymbol{z}_0^*, \boldsymbol{u}_0\right)$, depicted as the blue plane in Figure 3(a).



(a) Blue arrow denotes z linear approximation as the variation of u, red arrow represents Newton-Raphson 'infinitesimal' adjustment.



(b) Haptic-DMPs consist of classical DMPs, which is the curve on control space, generating control policy $u(t)$. Haptic-DMPs utilize Eq. (4.6) to compute the corresponding state $z(t)$ on EM. ϕ is the haptic cost in control space.

Figure 3. *Illustrations of Haptic-DMPs: the left figure illustrates the concept described in Equation (4.6); the right figure explains how Haptic-DMPs compute the control policy* $\mathbf{u}(t)$ *and the object's state* $\mathbf{z}(t)$ *simultaneously given an initial position* $\mathbf{u}_0$ *and a target position* $\mathbf{u}_T$*. Once the computation is complete, each policy returns a haptic cost, as defined in Equation (4.7).*

By combining both behaviors described in Equations (4.3) and (4.5), we can determine $z(t)$ using the following equation, ensuring that the state remains on the EM, which is represented by the curved surface in Figure 3(a).

$$d\boldsymbol{z} = -\left(\nabla^2_{zz}W\right)^{-1}\nabla^2_{uz}W d\boldsymbol{u} - \eta\left(\nabla^2_{zz}W\right)^{-1}\nabla_z W dt \tag{4.6}$$

In this equation, the first term aims to slide the state z in the tangent space with the EM, and the second term aims to pull z toward the EM in the normal direction.

Using Equation (4.6), the *Haptic-DMPs* are then introduced based on the following differential equations:

$$\tau\dot{\boldsymbol{u}} = \boldsymbol{v}, \tag{4.7a}$$

$$\tau\dot{\boldsymbol{v}} = \alpha_v(\beta_v(\boldsymbol{u}_T - \boldsymbol{u}) - \boldsymbol{v}) + \boldsymbol{f}(\boldsymbol{x}), \tag{4.7b}$$

$$\dot{\boldsymbol{z}} = -\left(\nabla^2_{zz}W\right)^{-1}\nabla^2_{uz}W\boldsymbol{v} - \eta\left(\nabla^2_{zz}W\right)^{-1}\nabla_z W, \tag{4.7c}$$

$$\dot{\phi} = \sqrt{\boldsymbol{v}^T\boldsymbol{G}^2_m(\boldsymbol{u})\boldsymbol{v}} \tag{4.7d}$$

with

$$\boldsymbol{G}_m(\boldsymbol{u}) \triangleq \nabla^2_{uu}W - \nabla^2_{uz}W\left(\nabla^2_{zz}W\right)^{-1}\nabla^2_{zu}W \tag{4.8}$$

Here, the first two equations—Equations (4.7a) and (4.7b)—correspond to conventional DMPs, but for the control input u, Equation (4.7c) resembles Equation (4.6) above. Equations (4.7c) and (4.8) involve computations of different gradients, including $\nabla^2_{zz}W(\boldsymbol{z},\boldsymbol{u})$, $\nabla^2_{uz}W(\boldsymbol{z},\boldsymbol{u})$, $\nabla_z W(\boldsymbol{z},\boldsymbol{u})$, and $\nabla^2_{uu}W(\boldsymbol{z},\boldsymbol{u})$. More specifically, we have

$$\nabla^2_{zz} W(\boldsymbol{z},\boldsymbol{u}) = \nabla^2_{zz} W_{ctrl}(\boldsymbol{z},\boldsymbol{u}) + \nabla^2 U(\boldsymbol{z}) \tag{4.9a}$$

$$\nabla^2_{\boldsymbol{uz}} W(\boldsymbol{z},\boldsymbol{u}) = \nabla^2_{\boldsymbol{uz}} W_{ctrl}(\boldsymbol{z},\boldsymbol{u}) \tag{4.9b}$$

$$\nabla_{z} W(\boldsymbol{z},\boldsymbol{u}) = \nabla_{z} W_{ctrl}(\boldsymbol{z},\boldsymbol{u}) + \nabla U(\boldsymbol{z}) \tag{4.9c}$$

$$\nabla^2_{\boldsymbol{uu}} W(\boldsymbol{z},\boldsymbol{u}) = \nabla^2_{\boldsymbol{uu}} W_{ctrl}(\boldsymbol{z},\boldsymbol{u}) \tag{4.9d}$$

The gradients related to $W_{ctrl}(\boldsymbol{z},\boldsymbol{u})$ can be calculated from its explicit function, such as Equation (2.3), while the gradients of $U(\boldsymbol{z})$ can be computed as discussed in Section 3.2. In Equation (4.7d), $\boldsymbol{G}_m(\boldsymbol{u})$ is the control Hessian, and ϕ is referred to as the haptic cost, which will be used in the optimization process later on. This haptic cost is equivalent to the accumulated control energy required along the trajectory when the robot follows a control policy $\boldsymbol{u}(t)$ (Campolo and Cardin, 2025). A higher ϕ value indicates that the robot requires greater control force during manipulation. For many manipulation tasks, the goal is to minimise this total control energy, making ϕ a suitable candidate for the cost function in our BBO algorithm, as presented next.

To learn the weights (parameters) $\boldsymbol{\Theta}$ of Haptic-DMPs, a BBO algorithm is employed (Stulp and Sigaud, 2013; Yang et al., 2023; Yang et al., 2025). As shown in Figure 2, in the i^{th} iteration, we perform R rollouts (indexed by r), where each rollout ($\boldsymbol{\Theta}_i^r$) involves sampling from a Gaussian distribution $\mathcal{N}$ as described in Equation (4.10). The distribution is centered around the optimal parameters $\boldsymbol{\Theta}^*_{i-1}$ obtained from the $(i-1)^{\text{th}}$ iteration

$$\boldsymbol{\Theta}_i^r = \mathcal{N}\left(\boldsymbol{\Theta}^*_{i-1}, \sigma_\Theta\right) \tag{4.10}$$

Here, σ_Θ denotes the variance of exploration. Once the new parameter $\boldsymbol{\Theta}_i^r$ for a rollout r is generated, Haptic-DMPs, as shown in Equation (4.7), compute a corresponding control policy $\boldsymbol{u}_i^r(t)$, given the initial pose $\boldsymbol{u}_0$ and the estimated target $\boldsymbol{u}_T$. The haptic cost ϕ_i^r for the rollout is then obtained. The parameter σ_Θ can also be interpreted as the exploration rate. A larger σ_Θ promotes broader exploration of $\boldsymbol{u}_i^r(t)$, while a smaller σ_Θ confines the exploration, keeping it closer to the optimal policy of the previous iteration. Subsequently, the BBO algorithm selects the rollout with the lowest haptic cost, which can be expressed as:

$$r^* = \arg\min_r \left(\phi^r\right) \tag{4.11}$$

Thus, the optimal parameter for the Haptic-DMP in this iteration is updated as $\boldsymbol{\Theta}_i^* = \boldsymbol{\Theta}_i^{r^*}$, which is called "**select and hold**" in Figure 2. This "**select and hold**" step also retains the minimal cost from the previous iteration to guide new explorations in a direction that aims to reduce the cost. This iterative process continues until ϕ_i^* converges.

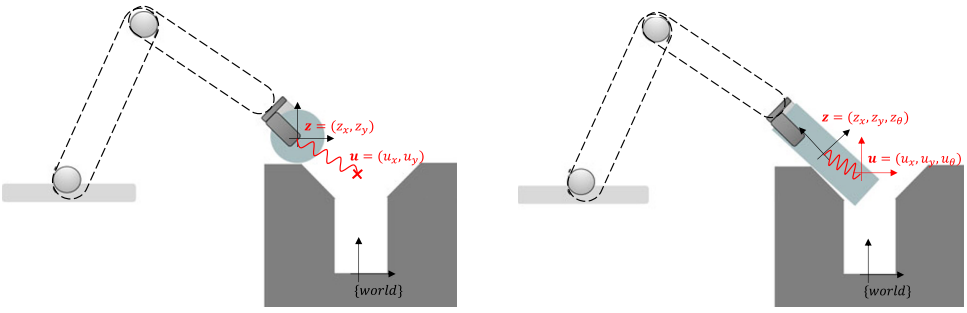
5. Case studies

In this section, we conduct two numerical examples (Figure 4) to verify our framework. The first example is a relatively straightforward task: disk-in-hole insertion. This is to aid the visualization of our results, as we want to show the regressed contact potential $U(\boldsymbol{z})$ in a three-dimensional plot, and hence the maximum dimension of $\boldsymbol{z}$ should be two. As such, we choose a centrally symmetric shape of an object where the contact force is independent of the orientation of the object. The second example is peg-in-hole insertion, which is a classical example in contact-rich tasks. The overall diagram of our proposed framework is shown in Figure 2, and the presentations of the two tasks in this section follow this framework.

5.1. Disk-in-hole manipulation

5.1.1. Model description

As shown in Figure 4(a), we define the world as x, y coordinate, one robot gripper grasps a disk $\boldsymbol{z}$ at its center. As the robot utilizes an impedance controller, there exists a desired position for the robot control $\boldsymbol{u}$.



(a) Manipulate a isotropic disk. (b) Manipulate an anisotropic peg.

Figure 4. *A robot manipulating an object into a hole. Our framework considers both indirectly controllable variable (state* $\mathbf{z}$*) and directly controllable variable (control input* $\mathbf{u}$*).*

Therefore, the configuration space is $\mathbb{R}^2 \times \mathbb{R}^2$, with $\boldsymbol{z} = (z_x, z_y)$ and $\boldsymbol{u} = (u_x, u_y)$. A virtual spring connects the robot control $\boldsymbol{u}$ and the disk $\boldsymbol{z}$ with stiffness $\boldsymbol{K}_0$. As such, the control energy $W_{ctrl}(\boldsymbol{z}, \boldsymbol{u})$ (Equation (2.3)) between the robot and the disk can be simplified as $W_{ctrl}(\boldsymbol{z}, \boldsymbol{u}) = \frac{1}{2}(\boldsymbol{u} - \boldsymbol{z})^T \boldsymbol{K}_0 (\boldsymbol{u} - \boldsymbol{z})$.

5.1.2. Data collection in Drake

In the Drake simulator (Tedrake et al., 2019), we uniformly distribute the variable $\boldsymbol{z}$ within a specified range that encompasses the entire hole, as illustrated in Figure 5. We record all pairs $(\boldsymbol{z}, \boldsymbol{F}(\boldsymbol{z}))$. For instances where the disk remains detached from the hole, the output force is a zero vector ($\boldsymbol{F} = \boldsymbol{0}$), indicating that the absence of contact results in negligible contact energy. Conversely, we catalogue the instances of contact along with the corresponding contact forces, which are then transformed into the world frame.

5.1.3. Regression using GPR with derivatives

After inputting the grid-like object positions $\boldsymbol{z}$ along with the corresponding output force, the dataset is collected. We then regress the entire environment using GPR (Equation (3.12)) with the kernel chosen as Gaussian (or exponentiated quadratic) defined as

$$\kappa(\boldsymbol{z}, \boldsymbol{z}') = \sigma_\kappa^2 \exp\left(-\frac{\|\boldsymbol{z} - \boldsymbol{z}'\|^2}{2\ell^2}\right) \tag{5.1}$$

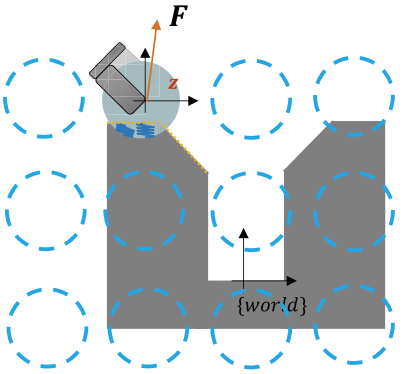
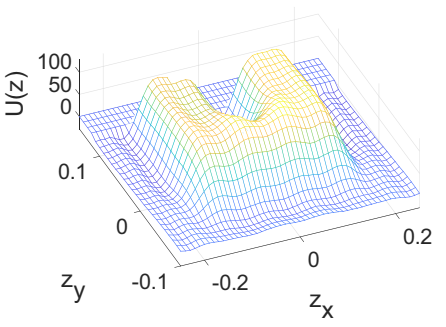
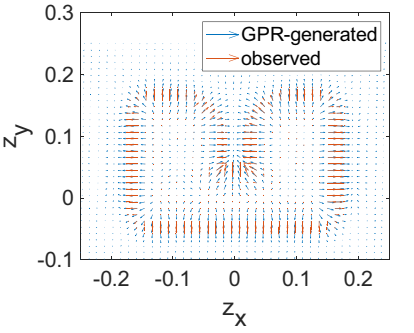


Figure 5. *A grid sampling to gather observation data. In simulation, the disk intersects with the hole to estimate contact force, where the contact is captured by a hydroelastic contact model (Tedrake et al., 2019). For the disk-in-hole task, we sample at different* (z_x, z_y).



(a) Regressed potential $U(\mathbf{z})$ from Eq. (3.9) on a grid map.



(b) Regressed gradient $\nabla_z U(\mathbf{z})$ from Eq. (3.10) v.s. observation.

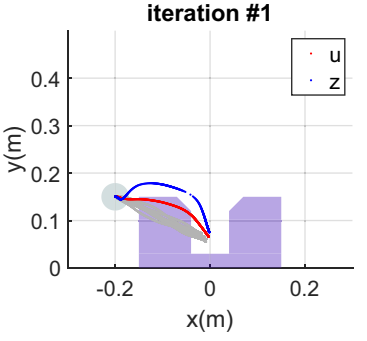
Figure 6. *Regressed potential $U(\mathbf{z})$ and gradient $\nabla_z U(\mathbf{z})$.*

where σ_κ is the amplitude coefficient, while ℓ denotes the kernel width. As depicted in Figure 6(a), the geometry of the contact potential resembles a hole, where the contact potential approaches zero in noncontact regions.

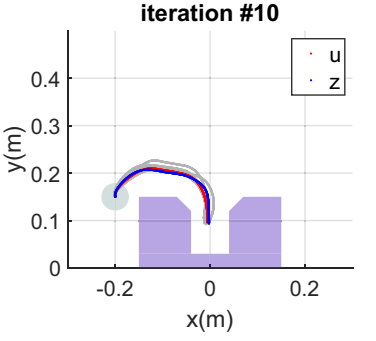
In Figure 6(b), we present the vector field obtained from both observations and regression. Given that the vectors represent physical forces, it is evident that the forces point outward at the boundary of the hole. Outside the hole, where no contact occurs, the observed forces are zero, while the regressed forces remain close to zero. However, due to the properties of Gaussian kernels and the sparsity of the observed data, the regressed force becomes relatively significant near the boundary, even in the absence of actual contact. This phenomenon can be mitigated by incorporating more observations, particularly in areas near the hole surface, for regression.

5.1.4. Trajectory planning using Haptic-DMPs and BBO

As shown in Figure 2, the control policy $\boldsymbol{u}(t)$ is parameterized by Haptic-DMPs (Equations (4.7a)–(4.7c)), and the cost function is ϕ as in Equation (4.7d). We set $R = 10$ rollouts for each iteration. We iterate the framework until the cost converges. The results are shown in Figures 7 and 8. The gray curves represent nonoptimal rollouts $\boldsymbol{u}^r$ during iteration i, the red curve indicates the optimal



(a) Iteration 1: the control policy mostly inside the hole so that the disk squeeze at the hole strongly.



(b) Iteration 10: the control policy avoid stuck at the hole where the optimal policy for insertion is perfect insertion.

Figure 7. *Iterations during BBO: the red curve denotes the optimal control policy $\mathbf{u}(t)$ in this iteration, the blue one denotes $\mathbf{z}(t)$, and the gray curves represent the nonoptimal explorations in each iteration of BBO.*

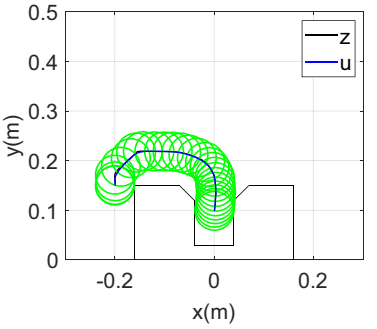


Figure 8. *The optimal policy* $\mathbf{u}(t)$ *implemented in Drake. For the disk-in-hole task, the optimal policy achieves a contactless insertion.*

policy in this iteration with the parameter $\boldsymbol{\Theta}_i^*$, and the blue curve shows the corresponding trajectory of the disk.

In the early iteration (Figure 7(a)), the control policy $\boldsymbol{u}(t)$ closely resembles a straight line toward the target. As a result, the cost is quite high, and the disk repeatedly makes contact with the boundary of the hole. However, after several iterations (Figure 7(b)), the BBO framework successfully identifies the optimal insertion policy, allowing the disk to be inserted into the hole with minimal or no contact, thereby significantly reducing the cost.

5.1.5. Verification in the simulator

After Haptic-DMPs compute the optimal control policy $\boldsymbol{u}(t)$, we implement this policy in the Drake simulator. The resulting trajectory of the disk is illustrated in Figure 8. The control policy $\boldsymbol{u}$ is shown in blue, and the simulated trajectory of the disk $\boldsymbol{z}(t)$ is shown in black. This test verifies the performance of our algorithm.

5.2. Peg-in-hole manipulation

For the peg-in-hole task shown in Figure 4(b), the variables are $\boldsymbol{z} = (z_x, z_y, z_\theta)$ and $\boldsymbol{u} = (u_x, u_y, u_\theta)$. The steps taken are the same as in the disk-in-hole example. We continue to use a grid map to sample the dataset, but with one more degree of freedom (DOF) for rotation.

5.2.1. Data collection and regression

Similar to the disk-in-hole task, we regress the contact potential $U(\boldsymbol{z})$ for the peg-in-hole task. Since the control $\boldsymbol{u} \in SE(2)$, we extract slices of u_θ at different angles to visualize the regressed potential function. From Figure 9, we observe that when the peg's rotation is fixed, the potential function resembles a hole

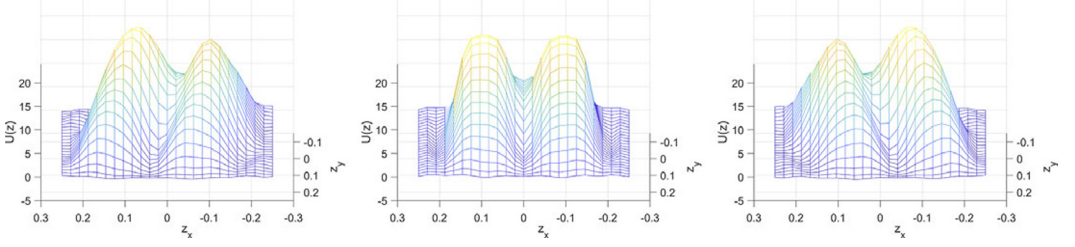
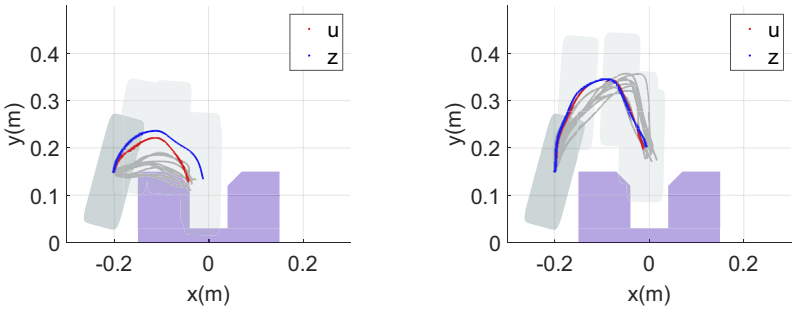


Figure 9. *A sliced observation of the regressed potential function* $U(\mathbf{z})$ *on a grid map.*



(a) Iteration 1: the initial policy mostly inside the hole so that the peg is squeezed at the hole strongly.

(b) Iteration 10: the control policy avoid unnecessary contact where the optimal policy slides along the chamfer.

Figure 10. *Iterations during BBO: the red curve denotes the optimal control policy* $\mathbf{u}(t)$ *in this iteration, the blue one denotes* $\mathbf{z}(t)$*, and the gray curves represent the nonoptimal explorations in each iteration of BBO.*

aligned with the peg's orientation. Specifically, when $z_\theta = 0^\circ$, $U(\boldsymbol{z})$ exhibits a hole-like shape analogous to the actual hole geometry. Additionally, the rotation of the peg does not affect the magnitude of the potential, as the contact potential is determined solely by the contact forces derived from the training data.

5.2.2. Trajectory planning using Haptic-DMPs and BBO

Similarly, we apply the BBO algorithm to the peg-in-hole example and plot the pose of the peg during each iteration to illustrate its rotation. In the early iteration (Figure 10(a)), the control policy starts as a straight-line trajectory, resulting in a high cost due to inefficient alignment and frequent contact with the hole's boundary. However, after several iterations (Figure 10(b)), the BBO algorithm adjusts the policy, guiding the peg away from the hole initially before smoothly inserting it, utilizing the chamfer to achieve insertion, significantly reducing the cost.

5.2.3. Verification in the simulator

The resulting trajectory of the peg simulated by Drake is illustrated in Figure 11. It can be seen that the peg occasionally makes contact with the hole's corners and slides along the chamfer to complete the insertion. This behavior occurs because, due to the sparsity of the observation data, the GPR provides a very smooth function approximation, which may not precisely capture sharp corners. Despite the approximation's imperfections, the task is still successfully completed with the help of impedance control, ensuring that the peg aligns correctly and completes the insertion.

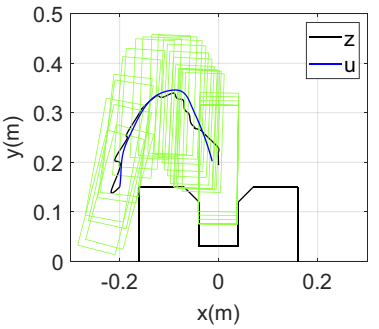


Figure 11. *Implement the optimal policy* $\mathbf{u}(t)$ *in the Drake. For the peg-in-hole task, the optimal policy slides along the chamfer to achieve the insertion task.*

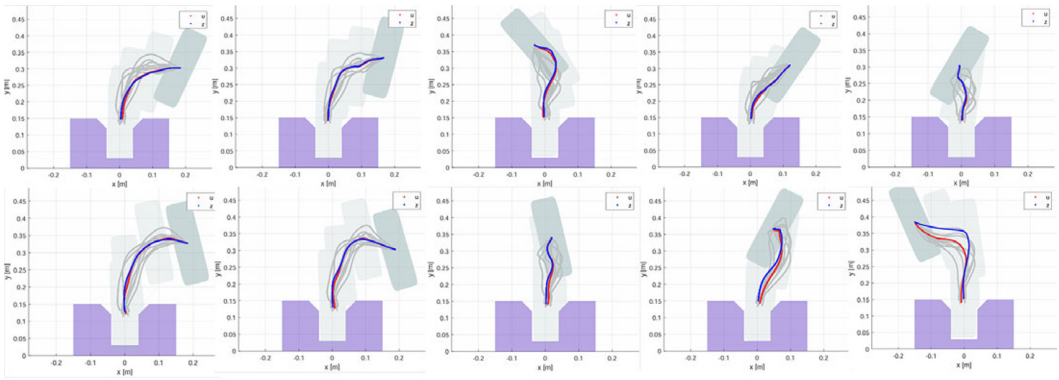


Figure 12. *Variation of the initial position of the peg* $\mathbf{z}_0$.

To validate the robustness of our framework, we also vary the initial position of the peg. We conducted 10 trials with randomly assigned initial positions, and all trials successfully inserted the peg into the hole as shown in Figure 12.

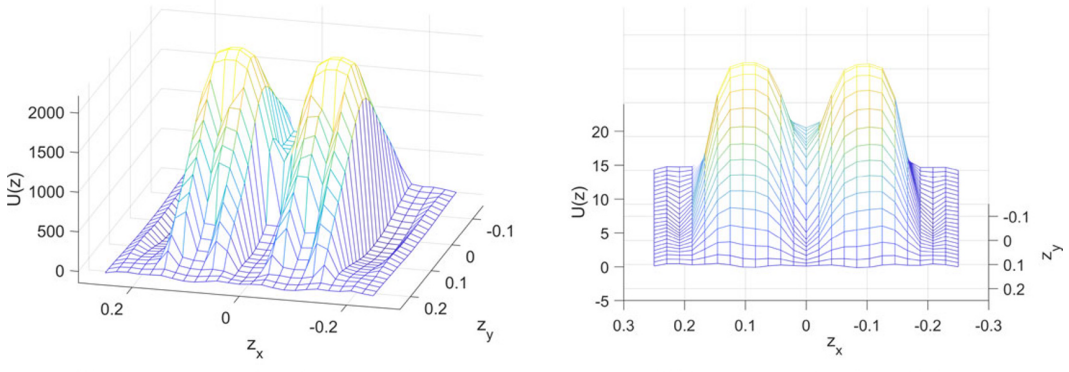
5.3. Further parameter analysis

As shown in Figure 2, our framework follows a block structure, allowing us to visualize and analyze each component separately.

5.3.1. Hyperparameters of GPR

The hyperparameters of GPR, such as the amplitude coefficient of the kernel σ_κ (as defined in Equation (5.1) and the variance of the noise σ_n in the observation (as defined in Equation (3.2)), influence the regressed contact potential $U(\boldsymbol{z})$. We vary the GPR hyperparameters and analyze how $U(\boldsymbol{z})$ changes as a result.

As observed in Figure 13, increasing σ_n has minimal impact on the results, whereas increasing σ_κ significantly affects the regressed potential. Despite these variations, the geometric shape of the hole remains consistent, although the absolute value of $U(\boldsymbol{z})$ changes. Additionally, the function becomes steeper, indicating increased contact stiffness or a stiffer object. Despite this, we test our framework with this parameter setting and find that the BBO successfully converges, discovering an insertion policy.



(a) Increase weight of σ_κ by a factor of 10.

(b) Increase σ_n by a factor of 100.

Figure 13. *Effect of varying GPR hyperparameters on the regressed contact potential* $U(\mathbf{z})$.

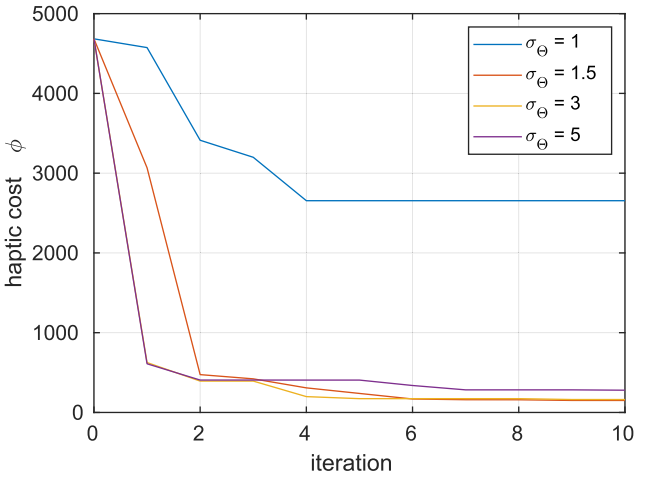


Figure 14. *Haptic cost ϕ with respect to iterations with different exploration rates.*

5.3.2. Exploration rate of BBO

To analyze the impact of the exploration rate σ_Θ (Equation (4.10)), we vary its value while keeping the initial position constant. The cost variation over iterations during the BBO process is shown in Figure 14.

- Smaller values of σ_Θ (e.g., 1): The cost decreases slowly and gets stuck in a local minimum. Physically, this corresponds to the peg getting stuck in front of the chamfer of the hole.
- Moderate values of σ_Θ (e.g., 1.5 and 3): A larger σ_Θ accelerates the cost reduction. Although $\sigma_\Theta = 3$ leads to a faster decrease than $\sigma_\Theta = 1.5$, both eventually converge to the same value, indicating a successful convergence of the BBO algorithm.
- Larger values of σ_Θ (e.g., 5): While the cost converges the fastest, the final converged cost is higher than that of $\sigma_\Theta = 1.5$ and $\sigma_\Theta = 3$, suggesting suboptimal performance due to excessive exploration.

Through these robust tests, which reveal the varying impact of different hyperparameters, the need for careful selection to optimize performance for specific tasks is again reiterated.

5.4. Further discussion

To better capture the environment in more detail, such as sharp corners, additional observation data are required. However, this introduces a trade-off with increased computational complexity due to larger datasets, as mentioned in Remark 3. This challenge can be addressed by employing scalable or incremental GPR methods.

In our simulations, Drake is used to collect force/torque data. In real-world scenarios, data collection can similarly be achieved by using a robot's end-effector to tap around its environment and record forces and torques with a force/torque sensor. An environment map can then be constructed using GPR with derivatives, and a control policy can subsequently be derived using Haptic-DMPs and BBO.

6. Conclusion

In this article, we have shown that by using only premeasured forces and torques (such as those from a simulator) as inputs for GPR with derivatives, it is possible to create an environment map from an energy-based perspective. This environmental energy is then employed in a mechanical framework operating under quasi-static conditions. Haptic-DMPs have been proposed to plan the trajectory for navigating the manipulation task in a contact-rich environment. Through numerical simulations of two classical

assembly tasks—disk-in-hole and peg-in-hole—it has been shown that the proposed method can find feasible paths to successfully perform these tasks. Our potential directions for further work involve incorporating prior knowledge about the environment into GPR, reducing computational complexity through scalable GPR techniques, and/or enabling incremental updates to GPR. These improvements would significantly improve the applicability of the framework in real-world experiments.

Data availability statement. The code is available at https://github.com/lllllyang/haptic_DMP.git and can also be found in (Nguyen et al., 2025).

Acknowledgments. This research is supported by the National Research Foundation, Singapore, under the NRF Medium-Sized Centre scheme (CARTIN). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not reflect the views of the National Research Foundation, Singapore.

Author contribution. Conceptualization: H.T.N., L.Y., and D.C. Methodology: H.T.N., L.Y., and D.C. Software: H.T.N. and L.Y. Validation: H.T.N. and L.Y. Formal analysis: H.T.N. and L.Y. Investigation: H.T.N. and L.Y. Resources: D.C. Data curation: H.T.N. and L.Y. Writing—original draft: H.T.N. and L.Y. Writing—review and editing: H.T.N., L.Y., C.L., and D.C. Visualization: H.T.N. and L.Y. Supervision: D.C. and C.L. Project administration: D.C. Funding acquisition: D.C. All authors have read and agreed to the published version of the manuscript.

Funding statement. This research was supported by grants from the National Research Foundation, Singapore, under the NRF Medium-Sized Centre scheme (CARTIN).

Competing interests. The authors declare none.

Ethical standard. The research meets all ethical guidelines, including adherence to the legal requirements of the study country.

References

Bhattacharjee T, **Shenoi AA**, **Park D**, **Rehg JM and Kemp CC** (*2015*) Combining tactile sensing and vision for rapid haptic mapping. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 1200–1207.

Caccavale F, **Natale C**, **Siciliano B and Villani L** (1999) Six-DOF impedance control based on angle/axis representations. *IEEE Transactions on Robotics and Automation 15*(2), 289–300.

Campolo D and Cardin F (2025) A geometric framework for quasi-static manipulation of a network of elastically connected rigid bodies. *Applied Mathematical Modelling 143*, 116003.

Campolo D and Cardin F (*2023*) Quasi-static mechanical manipulation as an optimal process. In *2023 62nd IEEE Conference on Decision and Control (CDC)*. IEEE, pp. 4753–4758.

Cross EJ, **Rogers TJ**, **Pitchforth DJ**, **Gibson SJ**, **Zhang S and Jones MR** (2024) A spectrum of physics-informed gaussian processes for regression in engineering. *Data-Centric Engineering 5*, e8.

Dattorro J (2010) *Convex Optimization & Euclidean Distance Geometry*. Lulu.com

Dawson-Elli AR and Adamczyk PG (2020) Design and validation of a lower-limb haptic rehabilitation robot. *IEEE Transactions on Neural Systems and Rehabilitation Engineering 28*(7), 1584–1594.

De Roos F, **Gessner A and Hennig P** (2021) High-dimensional Gaussian process inference with derivatives. In *International Conference on Machine Learning*. PMLR, pp. 2535–2545.

Diana M and Marescaux J (2015) Robotic surgery. *Journal of British Surgery 102*(2), e15–e28.

Ehsani K, **Han W**, **Herrasti A**, **VanderBilt E**, **Weihs L**, **Kolve E**, **Kembhavi A and Mottaghi R** (2021) Manipulathor: A framework for visual object manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4497–4506.

Gams A, **Nemec B**, **Ijspeert AJ and Ude A** (2014) Coupling movement primitives: Interaction with the environment and bimanual tasks. *IEEE Transactions on Robotics 30*(4), 816–830.

Garg S, **Sünderhauf N**, **Dayoub F**, **Morrison D**, **Cosgun A**, **Carneiro G**, **Wu Q**, **Chin T-J**, **Reid I**, **Gould S**, *et al.* (2020) Semantics for robotic mapping, perception and interaction: A survey. *Foundations and Trends® in Robotics 8*, 1, 1–2, 224.

Girão PS, **Ramos PMP**, **Postolache O and Pereira JMD** (2013) Tactile sensors for robotic applications. *Measurement 46*(3), 1257–1271.

Ijspeert AJ, **Nakanishi J**, **Hoffmann H**, **Pastor P and Schaal S** (2013) Dynamical movement primitives: Learning attractor models for motor behaviors. *Neural Computation 25*(2), 328–373.

Katayarna S, **Taniai T and Tanaka K** (2022) Quasistatic contact-rich manipulation via linear complementarity quadratic programming. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 203–210.

Katsura S, **Matsumoto Y and Ohnishi K** (2007) Modeling of force sensing and validation of disturbance observer for force control. *IEEE Transactions on Industrial Electronics 54*(1), 530–538.

Kazerouni IA, **Fitzgerald L**, **Dooly G and Toal D** (2022) A survey of state-of-the-art on visual SLAM. *Expert Systems with Applications 205*, 117734.

Kok M, **Solin A and Schön TB** (2024) Rao-blackwellized particle smoothing for simultaneous localization and mapping. *Data-Centric Engineering 5*, e15.

Koptev M, **Figueroa N and Billard A** (2021) Real-time self-collision avoidance in joint space for humanoid robots. *IEEE Robotics and Automation Letters 6*(2), 1240–1247.

LaValle SM (2006) *Planning Algorithms*. Cambridge University Press

Lee D-H, **Choi M-S**, **Park H**, **Jang G-R**, **Park J-H and Bae J-H** (2022) Peg-in-hole assembly with dual-arm robot and dexterous robot hands. *IEEE Robotics and Automation Letters 7*(4), 8566–8573.

Liu H, **Ong Y-S**, **Shen X and Cai J** (2020) When Gaussian process meets big data: A review of scalable gps. *IEEE Transactions on Neural Networks and Learning Systems 31*(11), 4405–4423.

Nguyen H-T, **Yang L**, **Lv C and Campolo D** (2025) Replication data for: Quasi-static contact-rich manipulation using dynamic movement primitives and haptic potential map. https://doi.org/10.5281/zenodo.15920444.

Saveriano M, **Abu-Dakka FJ**, **Kramberger A and Peternel L** (2023) Dynamic movement primitives in robotics: A tutorial survey. *The International Journal of Robotics Research 42*(13), 1133–1184.

Schneebeli HR and Wihler TP (2011) The Newton–Raphson method and adaptive ODE solvers. *Fractals 19*(01), 87–99.

Solak E, **Murray-Smith R**, **Leithead W**, **Leith D and Rasmussen C** (2002) Derivative observations in gaussian process models of dynamic systems. *Advances in Neural Information Processing Systems 15*.

Strub C, **Wörgötter F**, **Ritter H and Sandamirskaya Y** (2014) Using haptics to extract object shape from rotational manipulations. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 2179–2186.

Stulp F and Sigaud O (2013) Robot skill learning: From reinforcement learning to evolution strategies. *Paladyn, Journal of Behavioral Robotics 4*(1), 49–61.

Suomalainen M, **Karayiannidis Y and Kyrki V** (2022) A survey of robot manipulation in contact. *Robotics and Autonomous Systems 156*, 104224.

Suresh S, **Bauza M**, **Yu K-T**, **Mangelson JG**, **Rodriguez A and Kaess M** (*2021*) Tactile SLAM: Real-time inference of shape and pose from planar pushing. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 11322–11328.

Tedrake R, *et al.* (2019) Drake: Model-based design and verification for robotics.

Turlapati SH and Campolo D (2022) Towards haptic-based dual-arm manipulation. *Sensors 23*(1), 376.

Turlapati SH, **Gurnani J**, **Ariffin MZB**, **Kana S**, **Wong AHY**, **Han BS**, **Campolo D**, *et al.* (2024) Identification of intrinsic friction and torque ripple for a robotic joint with integrated torque sensors with application to wheel-bearing characterization. *Sensors (Basel, Switzerland) 24*(23), 7465.

Wang Y, **Beltran-Hernandez CC**, **Wan W and Harada K** (2022) An adaptive imitation learning framework for robotic complex contact-rich insertion tasks. *Frontiers in Robotics and AI 8*, 777363.

Whitney DE, *et al.* (1982) Quasi-static assembly of compliantly supported rigid parts. *Journal of Dynamic Systems, Measurement, and Control 104*(1), 65–77.

Williams CK and Rasmussen CE (2006) *Gaussian Processes for Machine Learning*, Vol. *2*. Cambridge, MA: MIT Press

Wirnshofer F, **Schmitt PS**, **Meister P**, **Wichert GV and Burgard W** (2019) State estimation in contact-rich manipulation. In *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 3790–3796.

Yang L, **Ariffin MZ**, **Lou B**, **Lv C and Campolo D** (2023) A planning framework for robotic insertion tasks via hydroelastic contact model. *Machines 11*(7), 741.

Yang L, **Nguyen H-T**, **Yu D**, **Lv C and Campolo D** (2025) Haptic rapidly-exploring random trees: A sampling-based planner for quasi-static manipulation tasks. *arXiv preprint arXiv:2506.00351*.

Yang L, **Turlapati SH**, **Lu Z**, **Lv C and Campolo D** (2025) A planning framework for stable robust multi-contact manipulation. *arXiv preprint arXiv:2504.02516*.

Yang L, **Nguyen H-T**, **Lv C**, **Campolo D and Cardin F** (2025) An energy-based numerical continuation approach for quasi-static mechanical manipulation. *Data-Centric Engineering 6*, e18.

Yang L, **Turlapati SH**, **Lv C and Campolo D** (2025) Planning for quasi-static manipulation tasks via an intrinsic haptic metric: A book insertion case study. *IEEE Robotics and Automation Letters 10*(6), 6111–6118.

Zhao H, **Chen Y**, **Li X and Ding H** (2023) Robotic peg-in-hole assembly based on reversible dynamic movement primitives and trajectory optimization. *Mechatronics 95*, 103054.

Cite this article: Nguyen H.-T, Yang L, Lv C and Campolo D (2025). Quasi-static contact-rich manipulation using dynamic movement primitives and haptic potential map. *Data-Centric Engineering*, 6, e41. doi:10.1017/dce.2025.10021