Ergodic Exploration using Tensor Train: Applications in Insertion Tasks

Suhan Shetty[1,2], João Silvério[1], and Sylvain Calinon[1,2]

***Abstract*—By generating control policies that create natural search behaviors in autonomous systems, ergodic control provides a principled solution to address tasks that require exploration. A large class of ergodic control algorithms relies on spectral analysis, which suffers from the curse of dimensionality, both in storage and computation. This drawback has prohibited the application of ergodic control in robot manipulation since it often requires exploration in state space with more than 2 dimensions. Indeed, the original ergodic control formulation will typically not allow exploratory behaviors to be generated for a complete 6D end-effector pose. In this paper, we propose a solution for ergodic exploration based on spectral analysis in multidimensional spaces using low-rank tensor approximation techniques. We rely on tensor train decomposition, a recent approach from multilinear algebra for low-rank approximation and efficient computation of multidimensional arrays. The proposed solution is efficient both computationally and storage-wise, hence making it suitable for its online implementation in robotic systems. The approach is applied to a peg-in-hole insertion task using a 7-axis Franka Emika Panda robot, where ergodic exploration allows the task to be achieved without requiring the use of force/torque sensors.**

***Index Terms*—Ergodic control, low-rank approximation, tensor methods, tensor train, tensor factorization, peg-in-hole insertion task, learning from demonstration.**

I. Introduction

AUTONOMOUS systems are often encountered with coverage tasks such as localization, tracking, and active learning. In such tasks, the agent might be required to explore a region of its state space, either due to the nature of the task at hand (e.g. surveillance) or due to uncertainties induced by sensory inaccuracies (e.g. peg-in-hole insertion). In such problems, the coverage task can be specified by a reference probability density function, which encodes the importance of exploration at any point of the state space. For such problems, a pattern-based coverage approach (e.g., a "lawnmower-type" strategy), as commonly used in low-dimensional state space, is not scalable, and hence not applicable to most of the applications encountered in practice [1]. Maximizing information gain, another popular approach to circumvent uncertainty, is not suitable for exploration since the coverage is likely to be concentrated in regions around information maxima disproportionately over the period of exploration [2].

Ergodic control provides an elegant solution to design control policies for such autonomous systems, in order to equip them with natural search behaviors. For a given reference probability density function over a domain of interest in the state space of the robot, a dynamical system is said to be ergodic if the fraction of time spent in a given region is proportional to the probability mass of that region [3]. This is formalized in ergodic theory, where the goal is to characterize how ergodic a given dynamical system is. Ergodic control, on the other hand, aims to design a control policy for a given autonomous system so that the trajectory evolution of the resulting dynamical system is ergodic for the reference probability distribution. Systems engineered in such a way have already found applications in robotics [2], [4]. The original approach to ergodic control, as proposed in [3], involves spectral analysis of the dynamical system evolution. This elegant method was proposed for point-mass systems having receding horizon control with infinitesimal control horizon. This original work paved the way for various extensions, with other types of dynamical systems and finite-horizon controllers [4], [2], [5]. The idea behind [3] is to minimize a metric, called the ergodic metric, that quantifies the match between the Fourier coefficients of a reference distribution and those of the time-averaged statistics of the system trajectory. As we will see in Section III-B, this method unfortunately suffers from the curse of dimensionality, prohibiting its applications to search spaces with more than 2 or 3 dimensions, which are often encountered in robot manipulation problems.

In this paper, we propose a solution to overcome this challenge by using low-rank tensor approximation techniques, in the form of a tensor train (TT), and hence expanding the domain of ergodic control to robot manipulation. Figure 1 shows an ergodic exploration behavior generated by the

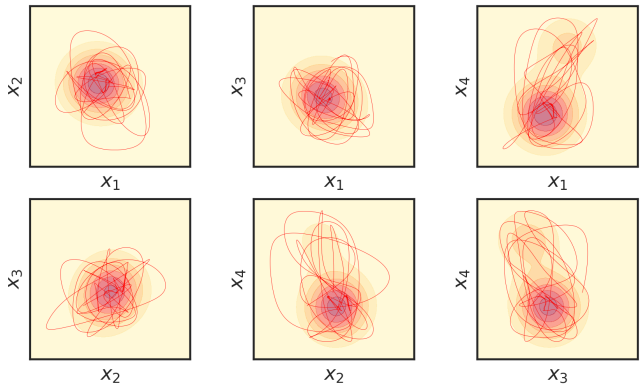


Fig. 1: The trajectory of a 4D point-mass system performing ergodic exploration using the proposed algorithm. The contour plot represents the reference distribution (taking here the form of a mixture of Gaussians).

[1] Idiap Research Institute, Martigny, Switzerland.
[2] EPFL, Lausanne, Switzerland.
E-mails: name.surname@idiap.ch
This work was supported by the CoLLaboratE project (https://collaborate-project.eu/), funded by the EU within H2020-DT-FOF-02-2018 under grant agreement 820767.

proposed method.

We showcase our approach in a 6D peg-in-hole insertion task using a robot manipulator, by considering the position and orientation of the end-effector[1]. In peg-in-hole scenarios, perception and modeling inaccuracies often compromise success, requiring the robot to leverage smart control strategies. Here, we propose to apply ergodic control to facilitate the insertion by letting the robot explore around the hole location in the 6D state space of the end-effector. In this application, we rely on human demonstrations to specify the distribution that the robot should use for an ergodic exploration.

The main contribution of this work is an ergodic control algorithm to generate exploratory behaviors in multi-dimensional spaces, which was previously considered to be an intractable problem. In particular, we improve the state of the art by proposing:

- Fast ways to compute the Fourier coefficients of multivariate functions, a well-known bottleneck in the ergodic control literature;
- The use of tensor train to exploit the inherent low-rank structure in the problem, which is used to overcome the curse of dimensionality in both real-time computation and storage requirements and facilitate implementation of ergodic control online on robotic systems.

The proposed ergodic control algorithm paves the way for two additional contributions in robotics. Particularly, we:

- Extend ergodic control to peg-in-hole tasks solved with an online policy, with a novel, principled and theoretically-grounded exploratory strategy for insertion tasks that does not rely on specialized sensors but only on human demonstrations;
- Provide a formal way to perform ergodic exploration in orientation by relying on the $\mathcal{S}^3$ Riemannian manifold.

To the best of our knowledge, this is the first time ergodic control is implemented online on a physical robot for exploration in dimension greater than 2. Note that the strategies to mitigate the curse of dimensionality introduced in this paper have the potential to be applied to many other applications in robotics to address real-time computation and limited storage requirements. Similarly, the proposed control strategy is not limited to manipulation applications, and can be extended to other robotics scenarios requiring high-dimensional coverage (e.g. 3D-object modeling, 6D surveillance).[2] Finally, even though we rely on human demonstrations to obtain the reference distribution, this reference can in practice be specified/learned in different ways (e.g. from sensor uncertainty models).

The paper is organized as follows: Section II gives a literature survey on ergodic control, tensor methods and control strategies for insertion tasks. Section III covers the necessary mathematical background to introduce our contribution. Particularly, Section III-A introduces tensor algebra. In Section III-B, the mathematical formulation of ergodic control is described, where the underlying challenges of the algorithm are outlined. Section III-C briefly introduces tensor decomposition techniques and Section III-D gives an overview of the tensor train decomposition, which is the main tool for low-rank approximation used in this paper. In Section IV, we propose low-rank approximation using the tensor train as a solution for multidimensional ergodic exploration. In Section V, we evaluate the proposed algorithm in simulation. Lastly, in Section VI we showcase the results of our approach in a peg-in-hole insertion task using a torque-controlled 7-axis Franka Emika Panda robot.

[1]A video of the experiment is available at https://sites.google.com/view/ergodic-exploration/

[2]The proposed algorithm has been numerically evaluated for state space of up to 15 dimensions.

II. Motivation and Related Work

A. Ergodic Control

Ergodic control was originally proposed by [3] in the form of a feedback control law designed for multi-agent systems, with an objective defined so that the agents trajectories cover a reference probability distribution. Here, the system considered is a point-mass system. The control policy is obtained by solving an optimization problem with an ergodic metric as the cost function (see Section III-B). The ergodic metric compares the Fourier series coefficients associated to the spatial reference distribution and the trajectory evolution of the system.

Although other possible choices of basis functions would be interesting to investigate (e.g. wavelets), the Fourier transform holds essential properties that are relevant to the considered problem. In [6], we discuss the use of Fourier series within ergodic control, including their links to cosine basis functions, and their properties for reference distributions in the form of mixtures of Gaussians.

The ergodic metric can be used as a starting point to design other forms of controllers. For example, ergodic controllers have been proposed using nonlinear dynamical systems and finite control horizons [2], using projection-based trajectory optimization [7], or using hybrid systems theory [5]. An overview of these methods with finite control horizon can be found in [8].

The main drawback of these methods is that they suffer from the curse of dimensionality (see Section III-B) when the dimension of the state space for ergodic exploration increases. This is due to the computational complexity and storage demanded in working with the ergodic metric and the control policy derived from it. Several authors deviated from the original ergodic control formulation to tackle this limitation, see e.g., [9], [10]. In [9] and [10], the authors relied on a different ergodic metric based on a Kullback-Leibler (KL) divergence measure for finite sensor footprint, where the control policy is obtained using sampling-based techniques. Sampling-based methods avoid the curse of dimensionality but they are still computationally expensive to address the real-time computational requirements of robotics systems. Moreover, it lacks the multi-scale coverage behavior [1], which is often essential for natural exploration.

This paper keeps the original methodology proposed by [3], which is the foundation of most literatures on ergodic control, and which has the advantage of providing closed form solutions for many of the commonly used models of dynamical

systems (kinematics-based) [4], while providing multi-scale coverage behaviour. To do so, we propose a solution based on a low-rank tensor approximation to overcome the curse of dimensionality. The proposed algorithm has intuitive hyper-parameters that can be adjusted to address the storage and computational constraints of the application.

Ergodic control can also be viewed from the perspective of a sampling method. In [11], the use of ergodic exploration is shown to be competitive to Markov chain Monte Carlo (MCMC) methods as a sampling technique. It is important to note that, unlike other sampling methods, the samples from ergodic control correspond to the trajectory of a dynamical system, thus providing a control policy that can be used in practice on robots for exploration behavior.

B. Tensor Methods

As we will see in Section III-B, the difficulty in ergodic control essentially arises from the storage and computation with multi-dimensional arrays involved in the algorithm. Tensor methods have recently gained popularity in signal processing, physics, applied mathematics, and machine learning communities due to their efficiency in storing and working with multidimensional arrays. These methods exploit the structure inherent in multidimensional arrays such as symmetry, parallel proportionality, and separability to represent them compactly and robustly. Furthermore, they allow performing algebraic operations efficiently in the compact format. Thus, the storage complexity and the algebraic operations complexity are significantly reduced. For a survey of classical tensor methods, we refer the readers to [12]. For applications of tensor methods in signal processing and machine learning, we refer to [13], [14]. In control, tensor methods have been used in [15] and [16] to solve multidimensional optimal control problems which were previously considered to be intractable.

C. Insertion Tasks

Peg-in-hole insertion is a typical and important problem in robotics. Many strategies to solve this problem depend on expensive force and torque sensors [17], [18]. Sensorless strategies [19], [20], [21], on the other hand, rely only on the state of the end-effector and provide a low cost solution. However, most of the sensorless strategies depend either on a predetermined trajectory that the robot end-effector needs to follow [21], or a full modeling of the insertion behavior [19], [20]. In [21], insertion is treated as a 3D (2D position and 1D orientation of the end-effector) trajectory tracking problem, where the reference trajectory for the robot end-effector is generated offline by using a coverage strategy such as ergodic control. As we will show, this strategy fails for the peg-in-hole insertion task considered in this paper, as a trajectory generated offline is often not possible to track due to obstructions from the surface of the hole and the peg. Our approach instead formulates the coverage problem in an online manner, with exploration simultaneously in the full 6D state space of the robot end-effector (3D position and 3D orientation).

In order to handle the exploration in orientation jointly with the position, we extend the control strategy to Riemannian manifolds by modeling the probability distribution of orientations, see [22], [23] for details. Subsequently, we use an online implementation of ergodic control as the solution for coverage. The algorithm proposed in this paper allows us to run the ergodic controller online on the robot for 6D insertion tasks.

TABLE I: Description of key notations and variables.

d	Dimension of the state space
K	Number of elementary basis functions along each dimension (K^d basis functions for the state space)
N	Degree of Gaussian quadrature rule ($N \in \mathbb{Z}^+$)
$\boldsymbol{j}, \boldsymbol{k}$	Tuple of indices
$\mathcal{K}$	Index set $\mathcal{K} = \{\boldsymbol{k} = (k_1, \ldots, k_d) : k_i \in \{1, \ldots, K\}\}$ with $K \in \mathbb{Z}^+$
i, j, k	Element of an index set (scalar)
$\boldsymbol{b}$	1D array (vector)
b_i	i-th element of $\boldsymbol{b}$
$\boldsymbol{A}$	2D array (matrix)
$\boldsymbol{\mathcal{G}}$	tensor of order > 2
$\boldsymbol{\mathcal{G}}_{\boldsymbol{k}}$	$\boldsymbol{k}$-th element of a tensor $\boldsymbol{\mathcal{G}}$ (a scalar)
$\boldsymbol{\mathcal{G}}^i$	i-th core (a third order tensor) of $\boldsymbol{\mathcal{G}}$ in its TT-format
$\boldsymbol{\mathcal{G}}^i_{:,:,k}$	k-th frontal slice (a matrix) of the third order tensor $\boldsymbol{\mathcal{G}}^i$
$\boldsymbol{r}$	$\boldsymbol{r} = (r_1, \ldots, r_{d-1})$ rank of a d-th order tensor in its TT format
r	Maximal rank of a tensor in its TT format, $r = \max(r_1, \ldots, r_{d-1})$
Ω	d-dimensional rectangular search space for ergodic exploration with $L > 0$: $\Omega = [0, L]^d$
$P(\boldsymbol{x})$	Reference probability distribution defined on Ω
$\boldsymbol{\mathcal{P}}$	Tensor formed by discretizing $P(\boldsymbol{x})$
$\phi_k(x)$	Elementary Fourier basis function for $[0, L]$, $\phi_k(x) = \cos(\frac{2\pi(k-1)x}{L})$
$\boldsymbol{\phi}(x)$	A vector of elementary Fourier basis functions for $[0, L]$, $\boldsymbol{\phi}(x) = (\phi_1(x), \ldots, \phi_K(x))$
$\boldsymbol{\Phi}(\boldsymbol{x})$	The Fourier basis functions for Ω (a d-th order tensor)
$C_t(\boldsymbol{x})$	Spatial statistics of the trajectory evolution $\boldsymbol{x}(t)$ of the dynamic system
$\xi(t)$	Ergodic metric
$\boldsymbol{\Lambda}$	Optimization weights in the ergodic metric
$\hat{\boldsymbol{\mathcal{W}}}$	Fourier series coefficients of $P(\boldsymbol{x})$
$\boldsymbol{\mathcal{W}}_t$	Fourier series coefficients of $C_t(\boldsymbol{x})$
$\boldsymbol{p}, \hat{\boldsymbol{p}}$	Robot current and desired end-effector position
$\boldsymbol{q}, \hat{\boldsymbol{q}}$	Robot current and desired end-effector orientation as a unit quaternion
$\boldsymbol{\mu}$	Mean demonstrated orientation in $\mathcal{S}^3$
$\boldsymbol{v}, \hat{\boldsymbol{v}}$	Robot current and desired end-effector orientation in the tangent space of $\boldsymbol{\mu}$
$\boldsymbol{\omega}$	Robot end-effector angular velocity
$\boldsymbol{K}_p, \boldsymbol{K}_d$	Impedance control stiffness and damping gains
$\hat{\boldsymbol{f}}, \hat{\boldsymbol{\tau}}$	Desired Cartesian wrench and joint torques for impedance control

III. Problem Definition and Background

In this section we lay out the mathematical background of our contribution. The notation used is summarized in Table I.

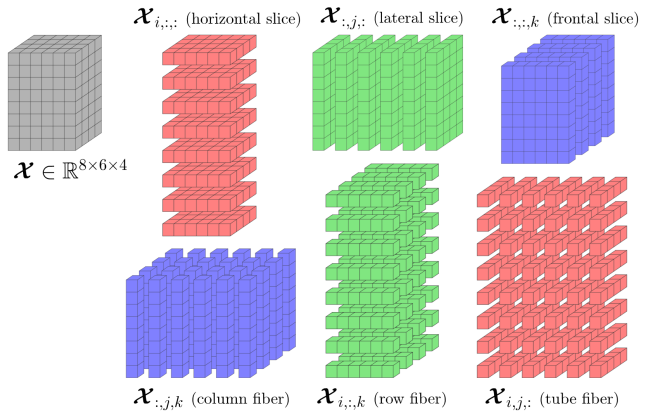


Fig. 2: The fibers and slices of a third order tensor $\boldsymbol{\mathcal{X}}$. They are the higher order analogs of rows and columns of a matrix.

A. Tensors

A tensor is a multidimensional array[3], whose order corresponds to the number of modes (or dimensions) of the array. A vector is a first order tensor and a matrix is a second order tensor. The $\boldsymbol{k}$-th element of a d-th order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{K_1\times\cdots\times K_d}$, with indices $\boldsymbol{k} = (k_1, \ldots, k_d)$ and $k_i \in \{1, \ldots, K_i\}$, is denoted by $\boldsymbol{\mathcal{X}}_{\boldsymbol{k}}$. Here, $K_i \in \mathbb{Z}^+$ represents the size of i-th mode, with $i \in \{1, \ldots, d\}$.

Fibers are the higher-order analogs of matrix rows and columns. A fiber is obtained by fixing every index but one (see Fig. 2, where ":" denotes the full list of entries for the corresponding mode).

The inner product of two tensors $\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}} \in \mathbb{R}^{K_1\times\cdots\times K_d}$ is given by

$$\langle \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}} \rangle = \sum_{\boldsymbol{k}\in\mathcal{K}} \boldsymbol{\mathcal{X}}_{\boldsymbol{k}} \boldsymbol{\mathcal{Y}}_{\boldsymbol{k}},$$

where $\mathcal{K} = \{\boldsymbol{k} = (k_1, \ldots, k_d) : k_i \in \{1, \ldots, K_i\}\}$. The Frobenius norm of a tensor $\boldsymbol{\mathcal{X}}$ is defined by

$$\|\boldsymbol{\mathcal{X}}\| = \langle \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{X}} \rangle^{\frac{1}{2}}.$$

The outer product of a tuple of one-dimensional arrays $(\boldsymbol{a}^1, \ldots, \boldsymbol{a}^d)$, with $\boldsymbol{a}^i = (a^i_1, \ldots, a^i_{K_i}) \in \mathbb{R}^{K_i}$ is denoted by $\boldsymbol{\mathcal{X}} = \boldsymbol{a}^1 \circ \cdots \circ \boldsymbol{a}^d \in \mathbb{R}^{K_1\times\cdots\times K_d}$, whose $\boldsymbol{k}$-th element is given by

$$\boldsymbol{\mathcal{X}}_{\boldsymbol{k}} = a^1_{k_1} \cdots a^d_{k_d}.$$

In a wide variety of applications, observed data could be represented naturally as tensors [12], [24]. However, in this article tensors arise either due to the discretization of an underlying multivariate function or from the tensor representation of the variables involved in the ergodic control algorithm. Even in the latter case, there will be an underlying multivariate function to evaluate elements of the tensor.

B. Ergodic Control

Ergodic control considers a point-mass dynamical system whose trajectory evolves such that its time-averaged statistics matches a desired reference probability distribution. In the method proposed originally in [3], the problem reduces to minimizing a cost function called ergodic metric, evaluating the distance between the Fourier coefficients of the reference distribution and that of the time-averaged statistics of the trajectory evolution of the dynamical system.

We assume a bounded d-dimensional rectangular domain: $\Omega = [0, L_1] \times \cdots \times [0, L_d]$ with $L_i > 0, \forall i \in \{1, \ldots, d\}$. Without loss of generality, we will consider $L_i = L, \forall i \in \{1, ..., d\}$. $\boldsymbol{x}(t) \in \mathbb{R}^d$ represents the trajectory of the dynamical system in the domain. The spatial statistics of the trajectory $\boldsymbol{x}(t)$ is defined as the fraction of time spent by the dynamical system at each point of the domain:

$$C_t(\boldsymbol{x}) = \frac{1}{t} \int_{\tau=0}^{t} \delta\big(\boldsymbol{x}(\tau) - \boldsymbol{x}\big) d\tau,$$

where δ is the Dirac delta function, and $\boldsymbol{x} = (x_1, \ldots, x_d) \in \Omega$ is a point in the domain.

Let $P(\boldsymbol{x})$ be the reference probability distribution for the exploration defined on Ω. The goal of ergodic control is to match the time-averaged spatial statistic $C_t(\boldsymbol{x})$ with the spatial distribution $P(\boldsymbol{x})$. The idea is to choose $K \in \mathcal{Z}^+$ orthonormal Fourier basis functions[4] satisfying the Neumann boundary conditions on the boundary of Ω: ϕ_k, $\forall k \in \{1, \ldots, K\}$, for each variable x, which is then organized as $\boldsymbol{\phi}(x) = (\phi_1(x), \ldots, \phi_K(x)) \in \mathbb{R}^K$. Although the results that follow apply to any such choice of basis function, we will use $\phi_k(x) = \cos(\frac{2\pi(k-1)x}{L})$ for numerical evaluation, see [6] for details. Then, orthonormal Fourier basis functions for Ω can be obtained by the elements of the d-th order tensor formed by the outer product of these vectors: $\boldsymbol{\Phi}(\boldsymbol{x}) = \boldsymbol{\phi}(x_1) \circ \cdots \circ \boldsymbol{\phi}(x_d) \in \mathbb{R}^{K\times\cdots\times K}$. With respect to this basis, the Fourier coefficients (cosine transforms) of $P(\boldsymbol{x})$ can be represented by a d-th order tensor $\hat{\boldsymbol{\mathcal{W}}}$. For a given index $\boldsymbol{k} = (k_1, \ldots, k_d) \in \mathcal{K}$, we have $\boldsymbol{\Phi}_{\boldsymbol{k}}(\boldsymbol{x}) = \phi_{k_1}(x_1) \cdots \phi_{k_d}(x_d)$, and $\hat{\boldsymbol{\mathcal{W}}}_{\boldsymbol{k}}$ represents the Fourier coefficient w.r.t. the basis $\boldsymbol{\Phi}_{\boldsymbol{k}}$, namely

$$\hat{\boldsymbol{\mathcal{W}}}_{\boldsymbol{k}} = \int_{x_1=0}^{L} \cdots \int_{x_d=0}^{L} P(\boldsymbol{x}) \boldsymbol{\Phi}_{\boldsymbol{k}}(\boldsymbol{x}) dx_1 \ldots dx_d. \tag{1}$$

The ergodic metric is then defined as

$$\xi(t) = \sum_{\boldsymbol{k}\in\mathcal{K}} \boldsymbol{\Lambda}_{\boldsymbol{k}} \big(\boldsymbol{\mathcal{W}}_{\boldsymbol{k}}(t) - \hat{\boldsymbol{\mathcal{W}}}_{\boldsymbol{k}}\big)^2, \tag{2}$$

where $\boldsymbol{\Lambda}_{\boldsymbol{k}} = (1 + \|\boldsymbol{k}\|^2)^{-\frac{d+1}{2}}$ are the weights for different frequencies, $\mathcal{K} = \{\boldsymbol{k} = (k_1, \ldots, k_d) : k_i \in \{1, \ldots, K\}\}$ and K is a sufficiently large positive integer. This way, higher priority is given to lower frequency contents of the reference distribution (i.e., exploration of large scale features), hence resulting in a multi-scale exploration behavior. $\boldsymbol{\mathcal{W}}(t)$ is the Fourier coefficients for the spatial statistics of the trajectory evolution $\boldsymbol{x}(t)$ at time t (i.e., of $C_t(\boldsymbol{x})$), which is given by

$$\boldsymbol{\mathcal{W}}(t) = \frac{1}{t} \int_{\tau=0}^{t} \boldsymbol{\Phi}(\boldsymbol{x}(\tau)) d\tau. \tag{3}$$

[3]The notion of tensors used in this paper is not to be confused with the one commonly used in physics as a multilinear map with specific properties. In this paper, a tensor is just a multidimensional array.

[4]In general, we can choose a different number of basis functions K_i for each dimension $i \in \{1, \ldots, d\}$. Without loss of generality, we will assume $K_i = K, \forall i$.

The ergodic control objective is $\lim_{t\to\infty} \xi(t) = 0$. For a dynamical system $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u})$ where $\boldsymbol{x} \in \mathbb{R}^d$, we want the ergodic dynamics w.r.t. the evolution of the states $\boldsymbol{x}(t) \in \Omega \subset R^d$. For infinitesimal control horizon, the solution for first-order systems is given as (see [3], [6] for details)

$$\begin{aligned} \dot{x}_i(t) &= \alpha \frac{b_i(t)}{\|\boldsymbol{b}(t)\|}, \\ \text{with} \quad b_i(t) &= \sum_{\boldsymbol{k}\in\mathcal{K}} \boldsymbol{\Lambda_k} \big(\boldsymbol{\mathcal{W}_k}(t) - \hat{\boldsymbol{\mathcal{W}}}_{\boldsymbol{k}}\big) \nabla_i \boldsymbol{\Phi_k}\big(\boldsymbol{x}(t)\big), \\ \boldsymbol{b} &= (b_1, \ldots, b_d), \end{aligned} \tag{4}$$

$$\nabla_i \boldsymbol{\Phi}\big(\boldsymbol{x}(t)\big) = \boldsymbol{\phi}(x_1) \circ \cdots \circ \frac{\partial \boldsymbol{\phi}(x_i)}{\partial x} \circ \cdots \circ \boldsymbol{\phi}(x_d), \tag{5}$$

where $\alpha > 0$ is a small real number.

For a fully actuated system with point-mass dynamics $\dot{\boldsymbol{x}} = \boldsymbol{u}$, with $\boldsymbol{u} = (u_1, \ldots, u_d)$ and maximum velocity $u_{\max}$, the control commands $u_i, i \in \{1, \ldots, d\}$ that minimize the ergodic metric are given by

$$u_i(t) = u_{\max} \frac{b_i(t)}{\|\boldsymbol{b}(t)\|}.$$

Similar expressions for control policies are available for other types of dynamical systems, such as second-order point-mass systems (acceleration command), or first order and second order Dubin's car models, see e.g. [3], [4]. The controller can also be extended to other nonlinear dynamic systems, see e.g. [8], [7], [5]. We demonstrate our approach using the simple point-mass system, as it captures the key challenges in scaling the ergodic control algorithm to higher-dimensional exploration, and because it remains a classical choice for ergodic exploration [11], [4], [25].

Algorithm 1 Ergodic control algorithm

Input: d, L, K, $u_{\max}$, T, and $P(\boldsymbol{x})$
Preprocessing:
Compute $\hat{\boldsymbol{\mathcal{W}}}$ (evaluate K^d multivariate integrals with (1))
Compute $\boldsymbol{\Lambda}$ (K^d function evaluations)
Initialise: $t = 0$, dt (time step), $\boldsymbol{x}(0)$, $\boldsymbol{\mathcal{W}}(0)$, $\boldsymbol{u}(0)$
{*Control Loop*}
while $t < T$ **do**
  $t \leftarrow t + dt$
  Update $\boldsymbol{x}(t)$
  Update $\boldsymbol{\mathcal{W}}(t)$ (use numerical integration)
  Compute $\nabla_i \boldsymbol{\Phi}\big(\boldsymbol{x}(t)\big), \forall i \in \{1, \ldots, d\}$
  for i=1,...,d **do**
    $b_i(t) = \sum_{\boldsymbol{k}\in\mathcal{K}} \boldsymbol{\Lambda_k} \big(\boldsymbol{\mathcal{W}_k}(t) - \hat{\boldsymbol{\mathcal{W}}}_{\boldsymbol{k}}\big) \nabla_i \boldsymbol{\Phi_k}\big(\boldsymbol{x}(t)\big)$
  end for
  Update $\boldsymbol{u}(t) = u_{\max} \frac{\boldsymbol{b}(t)}{\|\boldsymbol{b}(t)\|}$
end while
{*Note: In the control loop, each of the variables* $\hat{\boldsymbol{\mathcal{W}}}, \boldsymbol{\mathcal{W}}(t), \boldsymbol{\Lambda}, \nabla_i \boldsymbol{\Phi}\big(\boldsymbol{x}(t)\big)$ *need* $\mathcal{O}(K^d)$ *floating-point elements and each binary operation involving them has computational complexity* $\mathcal{O}(K^d)$.}

The ergodic control algorithm (see Algorithm 1) is simple but it suffers from the curse of dimensionality when the dimension d of the exploration domain Ω increases, which typically limits the use of the algorithm to problems of 2 or 3 dimensions. This is a drawback since many applications, in practice, are of dimensions $d > 3$. For example, the end-effector of a robot manipulator has 6 DOF (position and orientation). A naive implementation of the above algorithm for an exploration in the task space of a manipulator is then not feasible in practice.
The main challenges in the algorithm are listed below:

1) *Computation and storage of the Fourier series coefficients of the reference distribution* $\hat{\boldsymbol{\mathcal{W}}}$*:*
   This requires evaluation of the multidimensional integral in (1) K^d times to completely determine $\hat{\boldsymbol{\mathcal{W}}}$. Although this is a preprocessing step, the computational complexity involved and the storage requirement make it infeasible for engineering applications.
2) *Computation and storage of* $\boldsymbol{\Lambda}$*:*
   This is a preprocessing step. The computation of each element $\boldsymbol{\Lambda_k}$ is straightforward, but the number of function evaluations to find the complete tensor $\boldsymbol{\Lambda}$ and the required storage grow exponentially.
3) *Real-time implementation of the control loop:*
   The control loop will be very slow as the algebraic operations (such as addition, element-wise product, summation, Frobenius norm of tensors) are more time-consuming as the order of the tensors involved in computing the control policy increases. So, a realtime implementation of the control loop may not be possible.

To give an example of the computation time involved in ergodic control, we used the Python software implementation of ergodic control described in [3] [5]. By using $K = 10$ for the Fourier series coefficients and a spherical Gaussian of variance 0.01 at the center of Ω with $L = 1$ as the reference probability distribution, the preprocessing time to compute the coefficients $\hat{\boldsymbol{\mathcal{W}}}$ is approximately 16s in 2D, 5400s in 3D, and 2300000s in 4D ($d = 4$) with the multivariate integration (1) evaluated using the Python package *scipy.integrate.nquad*.[6] Note that the number of elements to be stored in these cases is 10^d, and for larger d (such as $d > 6$) it is highly likely that memory/storage requirement for each of the multidimensional arrays involved exceeds the limit. Moreover, the average time taken per control loop in the above setting for $d = 3, 4, 5, 6, 7$ are $0.0007s$, $0.001s$, $0.004s$, $0.06s$, and $0.8s$ correspondingly. As we will see next, the above-mentioned challenges can be solved using the capability of a state-of-the-art tensor decomposition technique called tensor train (TT). Using the proposed solution, for the above reference distribution, $\hat{\boldsymbol{\mathcal{W}}}$ can be represented using approximately Kd parameters and it can be computed in less than 0.002s and the average control loop takes less than 0.001s. Also, for other reference distributions and larger d, the pre-processing step and the control loop can be processed very fast.

[5]The configuration of computing system used is mentioned in Section V
[6]https://docs.scipy.org/doc/scipy/reference/tutorial/integrate.html

C. Tensor Decomposition Techniques

Tensor decomposition (or tensor factorization) extends matrix decomposition techniques (e.g. matrix singular value decomposition) to multidimensional arrays. They allow tensors to be encoded compactly using a set of lower-order and/or lower-rank tensors (called factors). These factor elements are operated, depending on the decomposition technique, using various algebraic operations to represent elements of the given tensor. The accuracy of the representation is often controlled by the *rank* of the tensor decomposition. The rank of a tensor has different meanings for different tensor decomposition techniques. If the tensor has some low-rank structure (often due to separability arising from symmetry or smoothness or parallel proportionality), the number of elements required to represent the given tensor using the decomposition techniques will be far fewer than the original tensor.[7] In addition to compactly representing tensors, tensor decomposition allows efficient application of algebraic operations in the compressed format.[8]

Popular tensor decomposition techniques include CANDECOMP/PARAFAC (CP), also known as canonical polyadic, Tucker, hierarchical Tucker (HT), and tensor train (TT), see [26] for a survey. For $d = 2$, all these techniques reduce to the well known singular value decomposition (SVD) of matrices. They can be interpreted as higher-order extensions of SVD, where these decomposition techniques differ for tensors of order $d > 2$. There exist many powerful algorithms to find the decomposition for each technique, most of them are based on the idea of alternating least squares (ALS) [12].

The TT decomposition [27] is a special case of HT decomposition. It shares the good properties of the Tucker decomposition because the space of the tensors in TT format (with a fixed maximal rank) forms a smooth manifold, thus allowing robust algorithms to determine the TT decomposition of a tensor. It also shares good properties with CP, as it scales well to high-order tensors. We exploit the tensor train (TT) decomposition in our approach, as it gathers the good properties of both CP and Tucker decomposition. For further details on the other tensor decomposition techniques, we refer to [12]. In the next section, a brief overview of TT decomposition is given.

D. Tensor Train Decomposition

A d-th order tensor $\boldsymbol{\mathcal{G}} \in \mathbb{R}^{K_1 \times \cdots \times K_d}$ in TT format is represented using a tuple of d third-order tensors $(\boldsymbol{\mathcal{G}}^1, \ldots, \boldsymbol{\mathcal{G}}^d)$. Here, $\boldsymbol{\mathcal{G}}^i \in \mathbb{R}^{r_{i-1} \times r_i \times K_i}$, $i \in \{2, \ldots, d-1\}$, $\boldsymbol{\mathcal{G}}^1 \in \mathbb{R}^{1 \times r_1 \times K_1}$ and $\boldsymbol{\mathcal{G}}^d \in \mathbb{R}^{r_{d-1} \times 1 \times K_d}$. As shown in Fig. 3,

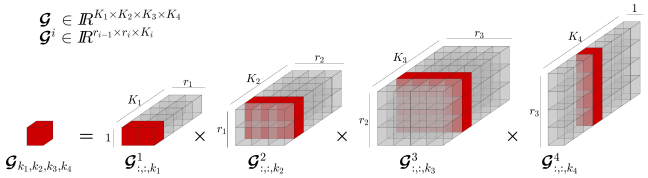


Fig. 3: An element of a tensor in TT-format can be accessed by multiplying the selected slices (matrices represented in red color) of the core tensors (factors). The figure depicts an example for a 4th order tensor $\boldsymbol{\mathcal{G}} \in \mathbb{R}^{5 \times 6 \times 7 \times 8}$ of rank $\boldsymbol{r} = (2, 3, 4)$.

the $\boldsymbol{k}$-th element, with $\boldsymbol{k} \in \mathcal{K} = \{(k_1, \ldots, k_d) \colon k_i \in \{1, \ldots, K_i\}, i \in \{1, \ldots, d\}\}$, is given by

$$\boldsymbol{\mathcal{G}}_{\boldsymbol{k}} = \boldsymbol{\mathcal{G}}^1_{:,:,k_1} \boldsymbol{\mathcal{G}}^2_{:,:,k_2} \cdots \boldsymbol{\mathcal{G}}^d_{:,:,k_d}, \tag{6}$$

where $\boldsymbol{\mathcal{G}}^i_{:,:,k_i} \in \mathbb{R}^{r_{i-1} \times r_i}$ represents the k_i-th frontal slice (a matrix) of the third order tensor $\boldsymbol{\mathcal{G}}^i$. The *TT-rank* of the tensor in TT representation is then defined as the tuple $\boldsymbol{r} = (r_1, r_2, \ldots, r_{d-1})$. We call $r = \max(r_1, \ldots, r_{d-1})$ as the *maximal TT rank*. For any given tensor, there always exists a TT decomposition (6).

The TT decomposition of a tensor can be considered as a particular way of writing the elements of a given tensor as a finite sum of separable products [26]. To explain this intuitively, consider a multivariate function $g \colon \Omega_g \subset \mathbb{R}^d \to \mathbb{R}$ such that $g(\boldsymbol{x}) = g(x_1, \ldots, x_d)$ which can be approximated as a finite sum of products of one dimensional functions ($g^i_{\hat{r}_i} : \mathbb{R} \to \mathbb{R}$, $i \in \{1, \ldots, d\}$ and $\hat{r}_i \in \{1, \ldots, r_i\}, r_i \in \mathbb{Z}^+$), namely

$$g(x_1, \ldots, x_d) \approx \sum_{\hat{r}_1=1}^{r_1} \cdots \sum_{\hat{r}_d=1}^{r_d} g^1_{\hat{r}_1}(x_1) \cdots g^d_{\hat{r}_d}(x_d). \tag{7}$$

Suppose we have a discretization of the domain $\Omega_g \subset \mathbb{R}^d$ given by the set $\{\boldsymbol{y} = (y_{k_1}, \ldots, y_{k_d}) : y_{k_i} \in \mathbb{R}, k_i \in \{1, \ldots, K_i\}, K_i \in \mathbb{Z}^+\}$ with the corresponding index set $\mathcal{K}$ as define above. Let $\boldsymbol{\mathcal{G}}$ be the tensor formed by evaluating the function g at these discretization points (i.e., $\boldsymbol{\mathcal{G}}_{\boldsymbol{k}} = g(y_{k_1}, \ldots, y_{k_d}), \forall \boldsymbol{k} = (k_1, \ldots, k_d) \in \mathcal{K}$). Then, the TT decomposition of $\boldsymbol{\mathcal{G}}$ is analogous to the representation (7) of the underlying multivariate function g, and it is given by

$$\boldsymbol{\mathcal{G}}_{\boldsymbol{k}} = \sum_{\hat{r}_1=1}^{r_1} \cdots \sum_{\hat{r}_d=1}^{r_d} \boldsymbol{\mathcal{G}}^1_{1,\hat{r}_1,k_1} \cdots \boldsymbol{\mathcal{G}}^d_{\hat{r}_{d-1},1,k_d}, \quad \forall \boldsymbol{k} \in \mathcal{K},$$

which is compactly written in (6) using matrix multiplication.

Let r be the maximal TT-rank and $K = \max(K_1, \ldots, K_d)$. Then, the number of elements in the TT representation is $\mathcal{O}(Kdr^2)$ as compared to $\mathcal{O}(K^d)$ elements in the original tensor. For small r, the representation is thus very efficient. In addition to storage efficiency, many numerical algebraic operations on tensors such as addition, scalar multiplication, Hadamard (or element wise) product, Frobenius norm can be

[7] For example, in matrix singular value decomposition, the low-rank corresponds to the case when many of the singular values are zero or negligible.

[8] This is analogous to algebraic operations on large matrices in their decomposition such as SVD, LU, etc. If we consider the right multiplication of a matrix $\boldsymbol{A} \in \mathbb{R}^{m \times s}$ with $\boldsymbol{B} \in \mathbb{R}^{s \times n}$, and if we have a decomposition of $\boldsymbol{A} = \boldsymbol{QR}$ with $\boldsymbol{R} \in \mathbb{R}^{p \times n}$ and $p \ll s$, then computing $\boldsymbol{C} = \boldsymbol{QRB}$ using the decomposition of $\boldsymbol{A}$ is more efficient than directly computing $\boldsymbol{C} = \boldsymbol{AB}$ using the raw $\boldsymbol{A}$. Similarly, for scalar multiplication of $\boldsymbol{A}$, computing $\boldsymbol{C} = (k\boldsymbol{Q})\boldsymbol{R}$ is more efficient than $\boldsymbol{C} = k\boldsymbol{A}$.

directly and efficiently applied in TT representation[9]. These operations are explained in [27], [28]. Most algebraic operations on tensors in TT format have computational complexity linear in d and K, and polynomial (often quadratic or cubic) in r. As we will see in Section IV, the proposed algorithm for ergodic control in this paper will exploit this efficiency in algebraic operation offered by TT format to speed up the computation in the control loop.

As explained above, the existence of low-rank structure (i.e., a low maximal TT-rank) of a given tensor is closely related to the separability of the underlying multivariate function. Although separability of functions is not a very well understood problem, it is known that smoothness[10] and symmetry of functions often induces better[11] separability of the functions. This has been exploited to solve PDEs and problems involving high-dimensional integration [26], and it will be used in this paper to solve the challenges in ergodic control.

Finding TT Decompositions

There exist many algorithms to find the TT decomposition of a tensor. The popular methods are TT-SVD, TT-DMRG, and TT-cross. TT-SVD and TT-DMRG, like matrix SVD, require the full tensor in memory to find the decomposition, and hence they are infeasible for higher order tensors. TT cross approximation (TT-cross) [29][30] is an extension of the cross approximation technique (also called CUR or skeleton decomposition) in matrix theory [31], see Fig. 4. It is appealing for many practical problems as it approximates the given tensor with controlled accuracy, by using only a small number of its elements. When applied to tensors, the cross approximation method needs access to only certain fibers of the original tensor at a time, and can hence works as a black box method.

Given a function (or a procedure) g that evaluates an element of a tensor $\boldsymbol{\mathcal{G}}$ given its index and an approximation accuracy ϵ in the Frobenius norm, TT-cross returns an approximate tensor in TT format $\hat{\boldsymbol{\mathcal{G}}} = \text{TT-cross}(g, \epsilon)$ to the tensor $\boldsymbol{\mathcal{G}}$ by querying only a portion of its elements. The expected error in approximation is less than the accuracy ϵ specified.

Thus, TT-cross avoids the need to store explicitly the original tensor, which may not be possible for higher order tensors. It can approximate a d-th order tensor, with maximal TT-rank r and size of each mode K, using only $\mathcal{O}(Kdr^2)$ samples and $\mathcal{O}(Kdr^3)$ operations (flops). This is very efficient if the TT-rank r of the tensor is low, which is typically the case in many engineering applications, including robotics. In this paper, we will use TT-cross to find TT decompositions.

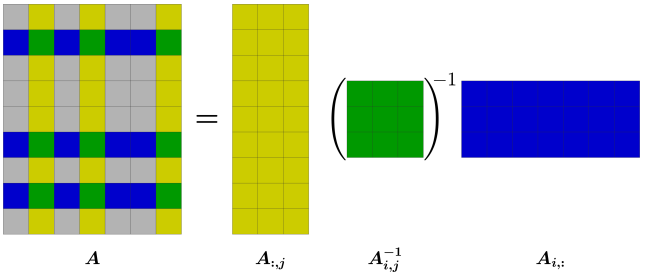


Fig. 4: TT-cross is an extension of skeleton decomposition in matrix theory. A rank-r matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ can be exactly recovered if we know r independent columns of $\boldsymbol{A}$ indexed by $\boldsymbol{j} = (j_1, \ldots, j_r)$, $\boldsymbol{A}_{:,\boldsymbol{j}} \in \mathbb{R}^{m\times r}$ and r independent rows of $\boldsymbol{A}$ indexed by $\boldsymbol{i} = (i_1, \ldots, i_r)$, $\boldsymbol{A}_{\boldsymbol{i},:} \in \mathbb{R}^{r\times n}$ of matrix $\boldsymbol{A}$, with their intersection $\boldsymbol{A}_{\boldsymbol{i},\boldsymbol{j}} \in \mathbb{R}^{r\times r}$ being nonsingular. Then, by skeleton decomposition we have $\boldsymbol{A} = \boldsymbol{A}_{:,\boldsymbol{j}}\boldsymbol{A}_{\boldsymbol{i},\boldsymbol{j}}^{-1}\boldsymbol{A}_{\boldsymbol{i},:}$.

TT-rounding

TT-rounding [27] is an important operation on a tensor in TT format. Most binary operations on tensors in TT format, although efficient, result in an increase in the TT-rank of the resultant tensor, where the resultant tensor is often not in its optimal TT representation. A repeated application of binary operations to a given tensor may result in an explosion of its TT-rank, which would effect the efficiency of subsequent operations on the tensor. For example, the addition of two tensors, both with TT-rank $\boldsymbol{r} = (r_1, \ldots, r_d)$, results in a tensor in TT format with rank $\boldsymbol{r} = (2r_1, \ldots, 2r_d)$. TT-rounding is an operation applied to tensors already in TT format to compress it to optimal TT representation and hence reduces its TT-rank. For a d-th order tensor $\boldsymbol{\mathcal{G}}$ in TT-format with maximal TT-rank r, TT-rounding has computational complexity $\mathcal{O}(Kdr^3)$. The TT-rounding procedure returns a tensor $\hat{\boldsymbol{\mathcal{G}}} = \text{TT-round}(\boldsymbol{\mathcal{G}}, \hat{r})$, for a given $\hat{r} < r$, such that its maximal TT-rank is less than $\hat{r}$ and the Frobenius norm of the residual $\|\hat{\boldsymbol{\mathcal{G}}} - \boldsymbol{\mathcal{G}}\|$ is as small as possible. Alternatively, we can specify an approximation accuracy ϵ to a tensor $\boldsymbol{\mathcal{G}}$ in TT format and the TT-rounding returns a tensor $\hat{\boldsymbol{\mathcal{G}}} = \text{TT-round}(\boldsymbol{\mathcal{G}}, \epsilon)$ with optimal TT-rank and $\|\hat{\boldsymbol{\mathcal{G}}} - \boldsymbol{\mathcal{G}}\| \leq \epsilon$.

IV. Algorithm Description

In this section, we give details of the solution proposed in this paper to overcome the challenges mentioned in Section III-B. Additionally, as part of our proposed strategy for 6D exploration in manipulation tasks, we propose a Riemannian manifold extension to allow ergodic exploration for orientation data represented as unit quaternions.

We use the TT representation for the variables involved in the algorithm, namely $\hat{\boldsymbol{\mathcal{W}}}, \boldsymbol{\mathcal{W}}(t), \boldsymbol{\Lambda}, \nabla_i \boldsymbol{\Phi}\big(\boldsymbol{x}(t)\big)$.[12] These variables, as described below, can be compactly represented in TT format, within a desired accuracy. Since all the operations

[9]For example, to gain an intuition about the efficiency of algebraic operations in TT format, consider two d-th order tensors of same shape $\boldsymbol{\mathcal{G}}$ and $\boldsymbol{\mathcal{H}}$ with known TT representations. The TT cores of the tensor $\boldsymbol{\mathcal{S}} = \boldsymbol{\mathcal{G}} + \boldsymbol{\mathcal{H}}$ can be directly obtained as $\boldsymbol{\mathcal{S}}^i_{:,:,k} = \text{diag}(\boldsymbol{\mathcal{G}}^i_{:,:,k}, \boldsymbol{\mathcal{H}}^i_{:,:,k})$ for $i \in \{2, \ldots, (d-1)\}$ and $\boldsymbol{\mathcal{S}}^j_{:,:,k} = \text{concatenate}(\boldsymbol{\mathcal{G}}^j_{:,:,k}, \boldsymbol{\mathcal{H}}^j_{:,:,k})$ for $j \in \{1, d\}$, where *diag* is the block diagonal matrix construction and *concatenate* represents vector concatenation operation. Also, the TT cores of the tensor formed by scalar multiplication $\boldsymbol{\mathcal{Q}} = c \cdot \boldsymbol{\mathcal{G}}$ can be obtained by multiplying the scalar only to the first core: $\boldsymbol{\mathcal{Q}}^1 = c \cdot \boldsymbol{\mathcal{G}}^1$ and $\boldsymbol{\mathcal{Q}}^i = \boldsymbol{\mathcal{G}}^i$ for $i \in \{2, \ldots, d\}$.

[10]By smoothness, we mean the degree of variation of the function across its domain. For example, a probability density function in the form of Gaussian mixture model (GMM) is considered to become less smooth as the number of mixture components (i.e., multi-modality) increases or the variance of the component Gaussians decreases (i.e., sharper peaks). More formally, the degree of smoothness can be defined using the properties of higher-order derivatives.

[11]By *better*, we mean fewer low-dimensional functions in the sum of products representation.

[12]$\nabla_i \boldsymbol{\Phi}\big(\boldsymbol{x}(t)\big), \forall i \in \{1, \ldots, d\}$ by its definition in (5) is a rank-1 TT. Its TT cores are $\big(\boldsymbol{\phi}(x_1), \ldots, \frac{\partial \boldsymbol{\phi}(x_i)}{\partial x}, \ldots, \boldsymbol{\phi}(x_d)\big)$.

to find the control policy, at each step of the control loop, are done on variables in TT format, the computational complexity is significantly reduced.

$\mathbf{\Lambda}$ can be found using the TT-cross approximation. Because of the involved symmetry (see the definition of $\mathbf{\Lambda}$), the resultant tensor in TT format is of very low rank. Maximal TT rank of 2 accurately captures the tensor with error less than 10^-2 in Frobenius norm and it can be computed in a fraction of a second (for d<15). We also represent $\boldsymbol{\mathcal{W}}(t)$ in TT format and use time integration in TT format [32] to compute it efficiently at each iteration of the control loop. Due to integration, as the TT-rank of this tensor may increase in an unbounded manner over time, we specify an upper bound (a hyperparameter) on its TT-rank. If other integration schemes are used, one can periodically use TT-rounding to cut off the TT-rank.

Finding $\hat{\boldsymbol{\mathcal{W}}}$ is a preprocessing step for the algorithm, and it is the most challenging part. Equation (1) shows that it requires the evaluation of a d-dimensional integral K^d times if implemented naively, namely

$$\hat{\boldsymbol{\mathcal{W}}}_{\boldsymbol{k}} = \int_{x_1=0}^{L} \cdots \int_{x_d=0}^{L} P(\boldsymbol{x}) \boldsymbol{\Phi}_{\boldsymbol{k}}(\boldsymbol{x}) dx_1 \cdots dx_d.$$

By using the properties of the TT format, the Fourier Coefficients can be computed in the following ways:

Method 1: By exploiting the properties of Gaussian mixture model (GMM) and the efficiency of TT-cross for reference distributions in the form of a GMM.

By using TT-cross the above integral needs to be computed only at the query points (Fourier coefficients) of the TT-cross, thus we obtain a significant savings in the number of evaluations of the multidimensional integral. This will require the integral in (1) to be evaluated $\mathcal{O}(Kdr^2)$ times and r is often small due to the structure of $\hat{\boldsymbol{\mathcal{W}}}$. However, computation of each multidimensional integral is still time consuming. For reference distributions in the form of a GMM, we have presented an analytical expression for the above integral in [6]. This avoids computation of the multi-dimensional integration using numerical schemes for distributions in the form of GMM for reasonably[13] large d as we use the analytical expression to compute the Fourier coefficients at the query points of TT-cross. Thus the method directly exploits the structure (i.e., smoothness) of the Fourier coefficients in $\hat{\mathcal{W}}$.

Method 2: By exploiting the properties of TT decomposition for arbitrary reference distributions.

We provide solution to the Fourier coefficients $\hat{\boldsymbol{\mathcal{W}}}$ without having to perform any multidimensional integration directly. By exploiting the properties of Gaussian quadrature rule for the integration of multivariate functions, we derive an analytical expression for the Fourier coefficients $\hat{\boldsymbol{\mathcal{W}}}$ in the TT representation for arbitrary functions. In this method, we exploit the smoothness of the reference probability distribution $P(\boldsymbol{x})$ to find $\hat{\boldsymbol{\mathcal{W}}}$ indirectly by evaluating $P(\boldsymbol{x})$ at a few points in its domain. This will be the method we use in the rest of the paper for finding $\hat{\boldsymbol{\mathcal{W}}}$ unless otherwise stated. This method to find Fourier coefficients is very efficient for smooth high-dimensional functions, which will be explained in the next section.

Note that, in practice, the reference distributions are smooth and/or the Fourier coefficients vary smoothly across their indices. Hence, a low-rank TT representation of $\hat{\boldsymbol{\mathcal{W}}}$ is expected.

A. Finding the Fourier Series Coefficients

In this section, we provide an analytical expression for the Fourier coefficients of a smooth but arbitrary distribution introduced in method 2 of Section IV. The proof is inspired by [33] where they have used separability structure in TT representation to find polynomial approximations of multivariate functions. A similar strategy has been used in [34] to evaluate high-dimensional integration of smooth functions. We use this strategy to find an analytical expression for the Fourier coefficients of functions directly in TT representation. The proof relies on quadrature rules (see Appendix A for the details) for numerical integration of multivariate functions. There are many possible options to choose the quadrature rule depending on the type of function $P(\boldsymbol{x})$ to be integrated. The following result applies to any choice of quadrature rule, however, for simplicity, we use Gaussian quadrature rule (G-Q) in this paper.

The idea is to find the TT representation $\boldsymbol{\mathcal{P}}$ of the multivariate function $P(\boldsymbol{x})$ evaluated at the discretization induced from the quadrature rule. Then, as we will see below, the TT representation of $\hat{\boldsymbol{\mathcal{W}}}$ can be obtained directly using $\boldsymbol{\mathcal{P}}$.

Let $x_j \in \mathbb{R}$ be the discretization points of the interval $[0, L]$ and α_j be the weights obtained from the quadrature rule, where $j \in \{1, \ldots, N\}$, with N representing the specified degree of approximation of the function. Then, we can discretize the domain Ω at $\boldsymbol{x_j} = (x_{j_1}, \ldots, x_{j_d})$, with $\boldsymbol{j} \in \mathcal{J}$, where $\mathcal{J} = \{\boldsymbol{j} = (j_1, \ldots, j_d) : j_i \in \{1, \ldots, N\}\}$ is the index set. Let $\boldsymbol{\mathcal{P}}$ be the tensor formed by evaluating the reference distribution $P(\boldsymbol{x})$ at the discretization points, so that $\boldsymbol{\mathcal{P}}_{\boldsymbol{j}} = P(\boldsymbol{x_j}), \forall \boldsymbol{j} \in \mathcal{J}$.

Let $(\boldsymbol{\mathcal{P}}^1, \boldsymbol{\mathcal{P}}^2, \ldots, \boldsymbol{\mathcal{P}}^d)$ be the TT cores of $\boldsymbol{\mathcal{P}}$ in its TT representation[14], so that for $\boldsymbol{j} = (j_1, \ldots, j_d) \in \mathcal{J}$ we have,

$$\boldsymbol{\mathcal{P}}_{\boldsymbol{j}} = \boldsymbol{\mathcal{P}}^1_{:,:,j_1} \boldsymbol{\mathcal{P}}^2_{:,:,j_2} \cdots \boldsymbol{\mathcal{P}}^d_{:,:,j_d},$$

then the TT cores of $\hat{\boldsymbol{\mathcal{W}}}$ are (see Appendix A for the proof)

$$\hat{\boldsymbol{\mathcal{W}}}^i_{:,:,k} = \sum_{j=1}^{N} \alpha_j \, \boldsymbol{\mathcal{P}}^i_{:,:,j} \, \phi_k(x_j), \quad \begin{matrix} \forall k \in \{1, \ldots, K\}, \\ \forall i \in \{1, \ldots, d\}, \end{matrix} \tag{8}$$

so that

$$\hat{\boldsymbol{\mathcal{W}}}_{\boldsymbol{k}} = \hat{\boldsymbol{\mathcal{W}}}^1_{:,:,k_1} \cdots \hat{\boldsymbol{\mathcal{W}}}^d_{:,:,k_d}, \ \forall \boldsymbol{k} \in \mathcal{K}.$$

Thus, we can compute the Fourier coefficients $\hat{\boldsymbol{\mathcal{W}}}$ by only investing in computing the TT decomposition $\boldsymbol{\mathcal{P}}$ of the discretized reference distribution. This can be done in $\mathcal{O}(Ndr^2)$ function evaluations of $P(\boldsymbol{x})$ using the TT-cross algorithm. The TT-rank of the tensor $\hat{\boldsymbol{\mathcal{W}}}$ will be same as that of the TT-rank of $\boldsymbol{\mathcal{P}}$. For a smooth reference distribution, $\boldsymbol{\mathcal{P}}$ will have

[13]Here, we consider d approximately up to 10 as the computation complexity of each Fourier coefficient using the analytical solution provided in [6] grows in proportion to 2^d. For details, refer to [6].

[14]This can be obtained using TT-cross: $\boldsymbol{\mathcal{P}} = \text{TT-Cross}(P(\boldsymbol{x}), \epsilon)$.

low TT-rank. This is a tremendous saving in computing the Fourier coefficients $\hat{\boldsymbol{\mathcal{W}}}$, and thus it overcomes the curse of dimensionality. The TT based algorithm for ergodic control is outlined in Algorithm 2.[15]

Algorithm 2 Ergodic Control using TT

Input: d, L, K, $u_{\max}$, T, and $P(\boldsymbol{x})$
Pre-Processing:
Compute $\boldsymbol{\mathcal{P}}$ using TT-cross
Find $\hat{\boldsymbol{\mathcal{W}}}$ using (8)
Compute $\boldsymbol{\Lambda}$ using TT-cross
TT-rounding of $\hat{\boldsymbol{\mathcal{W}}}$ (remove low-energy contents)
Initialise: dt (time step), $\boldsymbol{x}(0)$, $\boldsymbol{\mathcal{W}}(0)$ in TT format, $\boldsymbol{u}(0)$, and $t = 0$
{*Control Loop*}
while $t < T$ **do**
  $t \leftarrow t + dt$
  Update $\boldsymbol{x}(t)$ (time integration)
  Update $\boldsymbol{\mathcal{W}}(t)$ (Use TT time integration [32] with a fixed maximal TT-rank. Alternatively, use numerical integration such as Euler integration followed by TT-rounding)
  Compute $\nabla_i \boldsymbol{\Phi}\big(\boldsymbol{x}(t)\big), i \in \{1, \dots, d\}$(rank-1 TT)
  {*Using algebraic operations in TT*}
  for i=1,...,d **do**
    $b_i(t) = \sum_{\boldsymbol{k} \in \mathcal{K}} \boldsymbol{\Lambda_k} \big(\boldsymbol{\mathcal{W}_k}(t) - \hat{\boldsymbol{\mathcal{W}}}_{\boldsymbol{k}}\big) \nabla_i \boldsymbol{\Phi_k}\big(\boldsymbol{x}(t)\big)$
  end for
  Update $\boldsymbol{u}(t) = u_{\max} \frac{\boldsymbol{b}(t)}{\|\boldsymbol{b}(t)\|}$
end while
{*Note: In the control loop, the memory of each variable and the computational complexity of each algebraic operation has complexity grow linearly with d. Thus it avoids the curse of dimensionality. In particular, the computation of $b_i(t)$ requires a subtraction, a Hadamard product and an inner product involving tensors in TT format, hence it can be computed efficiently.* }

B. Ergodic Control on Riemannian Manifolds

Most manipulation tasks concern the full robot end-effector pose, which includes both its position and orientation. Hence, when designing exploration strategies for manipulation it is desirable to consider both. In the case of position, the ergodic control formulation in Section IV-A can be directly applied. However, since orientation data do not lie on a Euclidean space, exploration in orientation requires a special mathematical treatment. In this section, we extend ergodic control to handle data on a Riemannian manifold [22], [23], particularly the orientation manifold $\mathcal{S}^3$.

The orientation of the robot end-effector can be represented by a unit quaternion $\boldsymbol{q} \in \mathcal{S}^3$, comprised of a scalar part $q_s \in \mathbb{R}$ and a vector part $\boldsymbol{q}_v \in \mathbb{R}^3$ such that $\boldsymbol{q} = [q_s \ \boldsymbol{q}_v^\top]^\top$. For any point on the manifold $\boldsymbol{g} \in \mathcal{S}^3$ there exists a tangent space $\mathcal{T}_{\boldsymbol{g}}\mathcal{S}^3$ in which standard Euclidean methods can be applied to orientation. The function that maps a quaternion $\boldsymbol{q}$ from the manifold to a tangent space is called the *logarithmic map* and is given by

$$\mathrm{Log}(\boldsymbol{q}) = \begin{cases} \mathrm{acos}_*(q_s) \frac{\boldsymbol{q}_v}{\|\boldsymbol{q}_v\|}, & q_s \neq 1 \\ [0,0,0]^\top, & q_s = 1 \end{cases}, \tag{9}$$

where $\mathrm{acos}_*(\cdot)$ is a modified arc-cosine function [22]. Equation (9) maps $\boldsymbol{q}$ to the tangent space of the manifold origin. The mapping of $\boldsymbol{q}$ to the tangent space of an arbitrary point $\boldsymbol{g}$ is given by [22]

$$\mathrm{Log}_{\boldsymbol{g}}(\boldsymbol{q}) = \mathrm{Log}(\bar{\boldsymbol{g}} * \boldsymbol{q}), \tag{10}$$

where $\bar{(\cdot)}$ and $*$ denote the quaternion conjugate and product, respectively. The logarithmic map represents a unit quaternion $\boldsymbol{q}$ as a 3-dimensional Euclidean vector $\boldsymbol{v} \in \mathbb{R}^3$. Quaternions can be retrieved from the tangent space through the *exponential map*

$$\mathrm{Exp}(\boldsymbol{v}) = \begin{cases} \left[\cos(\|\boldsymbol{v}\|), \ \sin(\|\boldsymbol{v}\|) \frac{\boldsymbol{v}^\top}{\|\boldsymbol{v}\|}\right]^\top, & \|\boldsymbol{v}\| \neq 0 \\ [1,0,0,0]^\top, & \|\boldsymbol{v}\| = 0 \end{cases}, \tag{11}$$

which, analogously to (10), maps from an arbitrary tangent space $\mathcal{T}_{\boldsymbol{g}}\mathcal{S}^3$ back to the manifold through

$$\mathrm{Exp}_{\boldsymbol{g}}(\boldsymbol{v}) = \boldsymbol{g} * \mathrm{Exp}(\boldsymbol{v}). \tag{12}$$

Given a set of unit quaternions (e.g. obtained from demonstrations), we formulate orientation-ergodic control by modeling their distribution in the tangent space of their mean. For M end-effector orientations $\{\boldsymbol{q}_i\}_{i=1}^M$, the mean on the manifold $\boldsymbol{\mu} \in \mathcal{S}^3$ is computed iteratively with (see [22], [23] for details)

$$\boldsymbol{v} = \frac{1}{M} \sum_{i=1}^{M} \mathrm{Log}_{\boldsymbol{\mu}}(\boldsymbol{q}_i), \qquad \boldsymbol{\mu} \leftarrow \mathrm{Exp}_{\boldsymbol{\mu}}(\boldsymbol{v}). \tag{13}$$

All quaternions in the dataset can thus be mapped to the tangent space of the mean through $\{\boldsymbol{v}_i\}_{i=1}^M = \{\mathrm{Log}_{\boldsymbol{\mu}}(\boldsymbol{q}_i)\}_{i=1}^M$, allowing the proposed ergodic exploration (Algorithm 2) to be performed in $\mathbb{R}^3$, even for orientation. As desired orientations are computed, in the tangent space, at each time step by $\hat{\boldsymbol{v}}(t) = \boldsymbol{v}(t) + \boldsymbol{u}(t)dt$, the exponential map generates a desired unit quaternion for the robot to track, using

$$\hat{\boldsymbol{q}}(t) = \mathrm{Exp}_{\boldsymbol{\mu}}(\hat{\boldsymbol{v}}(t)). \tag{14}$$

In this way, ergodic control for end-effector poses is formulated as a 6D problem, where the first 3 dimensions correspond to position and the last 3 to orientation.

V. Numerical Evaluation

In this section, we demonstrate the computational efficiency of the TT-based algorithm for ergodic control through simulations. The simulations are performed on a Lenovo Thinkpad personal computer with Intel(R) Core(TM) i7-8565U CPU at 1.80GHz with 24GB RAM. We use ttpy, a Python-based toolbox for working with TT.[16]

In general, for $d > 5$, even for GMM as reference probability distribution, method 2 is observed to be faster than

[15]Our implementation of Algorithm 2 in Python can be found at https://sites.google.com/view/ergodic-exploration/.

[16]https://github.com/oseledets/ttpy

			Number of components in GMM						Uniform distribution	
			2		4		6			
			With TT	Without TT	With TT	Without TT	With TT	Without TT	With TT	Without TT
5D	Number of parameters:	$\nabla_i \mathbf{\Phi}$	50	10^5	50	10^5	50	10^5	50	10^5
		$\mathbf{\Lambda}$	160	10^5	160	10^5	160	10^5	160	10^5
		$\hat{\boldsymbol{\mathcal{W}}}$	160	10^5	548	10^5	1032	10^5	50	10^5
	Average time per control loop:		0.002	0.004	0.003	0.004	0.0036	0.004	0.002	0.004
	Pre-processing time:		0.2	–	0.87	–	2.4	–	0.033	–
6D	Number of parameters:	$\nabla_i \mathbf{\Phi}$	60	10^6	60	10^6	60	10^6	60	10^6
		$\mathbf{\Lambda}$	200	10^6	200	10^6	200	10^6	200	10^6
		$\hat{\boldsymbol{\mathcal{W}}}$	200	10^6	695	10^6	1431	10^6	60	10^6
	Average time per control loop:		0.0032	0.063	0.0046	0.063	0.0054	0.063	0.003	0.063
	Pre-processing time:		0.03	–	1.26	–	3.6	–	0.04	–
7D	Number of parameters:	$\nabla_i \mathbf{\Phi}$	70	10^7	70	10^7	70	10^7	70	10^7
		$\mathbf{\Lambda}$	240	10^7	240	10^7	240	10^7	240	10^7
		$\hat{\boldsymbol{\mathcal{W}}}$	233	10^7	860	10^7	1801	10^7	70	10^7
	Average time per control loop:		0.0048	0.8	0.0068	0.8	0.0075	0.8	0.0036	0.8
	Pre-processing time:		0.035	–	1.53	–	4.9	–	0.043	–

TABLE II: Computational speed and storage requirements in Ergodic control for different reference probability distributions with $K = 10$ and $L = 1$m. The components of GMM are spherical Gaussians with 0.005 variance. All the tensors in TT format are approximated with an accuracy of 10^{-2} in the Frobenius norm. The preprocessing time for the naive approach (without using TT) is not given in the table as it is computationally infeasible in the computing device used for the experiment.

method 1 for smooth distributions (GMM without too many sharp peaks)[17]. However, for GMM with very sharp peaks and $d < 10$, one may prefer method 1 as it relies directly on the structure of $\hat{\boldsymbol{\mathcal{W}}}$ and hence it could be faster for such reference distributions. In the remaining part of this section, we evaluate method 2 given in Section IV.

A naive implementation without using tensor decomposition techniques would require K^d elements to store each of the tensors: $\hat{\boldsymbol{\mathcal{W}}}$, $\mathbf{\Lambda}$, and $\nabla_i \mathbf{\Phi}(\boldsymbol{x})$ $(i \in \{1, \dots, d\})$. However, a TT representation requires less than $4Kd$ elements for $\mathbf{\Lambda}$ (with approximation error 10^{-2} in the Frobenius norm) and only Kd elements to exactly represent $\nabla_i \mathbf{\Phi}(\boldsymbol{x})$. As $\nabla_i \mathbf{\Phi}(\boldsymbol{x})$ is a rank-1 tensor with explicit analytical expressions for its TT cores, it can be computed very fast. Computing the weights $\mathbf{\Lambda}$ can be done in a fraction of a second for $d \leq 15$.

The computation and storage of $\hat{\boldsymbol{\mathcal{W}}}$ depends on the smoothness of the reference probability distribution. For the evaluation, we define our reference distribution as an isotropic Gaussian distribution at the centre of the domain with variance 0.015, where we used $L = 1$, $K = 5$, and $N = 10$. As can be seen in Fig. 6, the time taken to compute $\hat{\boldsymbol{\mathcal{W}}}$ grows approximately linearly with d, and it is less than a second for the chosen reference distribution with $d \leq 10$. After we find $\hat{\boldsymbol{\mathcal{W}}}$ and $\mathbf{\Lambda}$ in TT-format, we process them using TT-rounding with an accuracy of $\epsilon = 10^{-2}$. In the proposed algorithm, as can be seen in Fig. 5, the average time taken per control loop increases almost linearly with the number of dimensions d, whereas with a naive implementation (without using TT) the time taken in the control loop grows exponentially with d.

[17]This is due to the computationally complexity involved in evaluating the analytical expression for each Fourier series coefficient. It grows in proportion to 2^d and linearly with the number of mixture components in the GMM. For details, see [6].

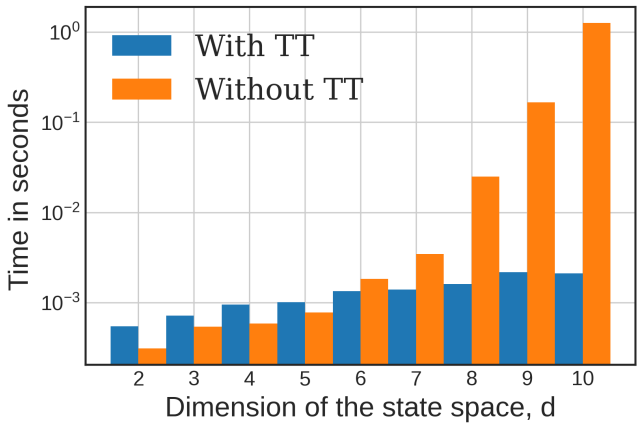


Fig. 5: Average time taken (in log-10 scale) per control loop of the ergodic control algorithm with $K = 5$, $l = 1$, and a reference distribution in the form of an isotropic Gaussian with variance 0.015. The exponential growth in the computational complexity can be observed for the standard approach (without using tensor decomposition), whereas the proposed approach avoids the curse of dimensionality.

The trend remains the same for other reference distributions, see Table II. In practice, to find $\boldsymbol{\mathcal{P}}$, $N = 10$ with TT-cross accuracy of 10^{-2} is observed to be sufficient[18]. For distributions with multiple modes and sharp peaks, a larger N should be used for an accurate representation of $\boldsymbol{\mathcal{P}}$, which directly yields $\hat{\boldsymbol{\mathcal{W}}}$ by (8).

The computation of $\boldsymbol{\mathcal{W}}(t)$ in the control loop requires some attention. At each iteration of the control loop, the rank of the

[18]A metric that can be used to verify if the error in approximation $\boldsymbol{\mathcal{P}} = \text{TT-Cross}(P, \epsilon)$ is reasonable is through the probability mass in Ω. In the numerical evaluation, indeed, the probability mass evaluated through the TT-representation $\boldsymbol{\mathcal{P}}$ is found to be nearly (proportional to the error ϵ specified) equal to that of actual mass of P in Ω (which we often design it to be equal to 1).

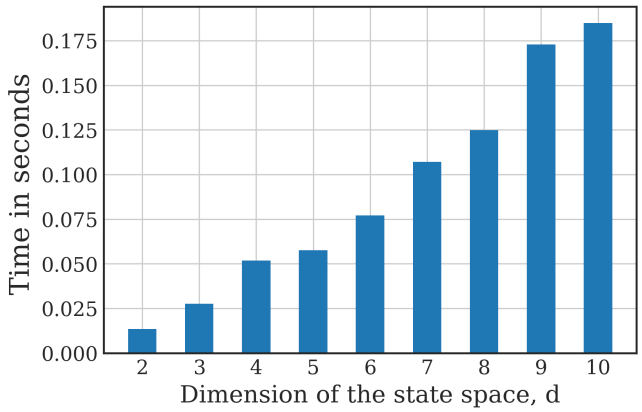


Fig. 6: Time taken (linear scale) to compute the Fourier coefficients $\hat{\mathcal{W}}$ using the proposed technique for $K = 5$, $l = 1$, and a reference distribution in the form of an isotropic Gaussian with variance 0.015. It can be observed that the computational complexity tends to grow linearly with d.

tensor $\mathcal{W}(t)$ may keep increasing due to the integration, see (3). This could be a problem if the time period of ergodic exploration is very high. So, it is necessary to upper bound the TT-rank of this tensor using TT-rounding with a specified maximal TT-rank. Setting an excessively low value for the maximal rank may lead to convergence issues and specifying a large value for maximal rank slows down the speed of computation of each control loop. Thus, this hyperparameter must be chosen carefully. In the numerical evaluation, our experiments revealed that a maximal rank of dr for $\mathcal{W}(t)$, where r is maximal TT-rank of $\hat{\mathcal{W}}$, worked well for $d < 15$.

The TT-cross algorithm to compute $\mathcal{P}$ needs only the approximation accuracy of ϵ, and the TT-rank r of the resultant tensor $\mathcal{P}$ is determined by the specified accuracy ϵ. An approximation accuracy of $\epsilon = 10^{-2}$ is found to be sufficient in practice for ergodic control. After finding $\hat{\mathcal{W}}$ using (8), a TT-rounding operation with accuracy $\epsilon = 10^{-2}$ significantly reduces the TT-rank of the resultant tensor and hence significantly speeds up the computation in the control loop. As the number of dimensions d increases ($d > 10$), $K \leq 5$ is observed to be sufficient to obtain good exploratory behavior. In all the test cases, the ergodic metric (2) converges to zero over time with the proposed algorithm, and for sufficiently large number of samples, the mean of the samples of ergodic exploration matches that of the mean of the reference probability distribution.

The TT representation allows compact representation of the tensors involved and the storage complexity also grows linearly with d. These properties allow our algorithm to be implemented for real-time applications on devices with small memory and limited computational power, which are often the case in robotic systems. Another important property of our algorithm is that the approximation of the tensors involved such as $\hat{\mathcal{W}}$, $\mathcal{W}(t)$ and $\mathbf{\Lambda}$ can be controlled precisely using TT-rounding. In practice, TT-rounding with accuracy 10^{-2} is observed to be sufficient for all practical purposes considered here. Furthermore, doing these approximations in the spectral domain results in negligible impact on the time domain behaviour of the system, thanks to Parseval's theorem (it can be considered as a noise filter in ergodic motion). This also provides a convenient trade-off between accuracy of approximation in ergodic exploration and the speed of computation in the control loop.

VI. Experiment: Sensorless Peg-in-Hole Insertion using Ergodic Exploration

We evaluate the proposed approach in an insertion task. We formulate the insertion task as a 6D exploration problem where we simultaneously address the uncertainty about the insertion pose in position (3D) and orientation (3D) in the robot task space. Our method is well suited for peg-in-hole insertion tasks where uncertainties may arise from several sources, including variable grasps of the peg, unprecise locations of the hole, and unmodeled manufacturing defects of the involved components (gripper fingers, pegs and holes). In the considered experiment, the reference probability distribution for exploration is found using information from human demonstrations. The human demonstrates the key regions for exploration in the state space of the end-effector and we use a Gaussian mixture model (GMM) to model the reference probability distribution based on the datapoints collected during the demonstrations. As a means to intuitively show the effectiveness of our approach, we begin by comparing it to three baselines of exploratory behaviors commonly used in insertion tasks. For this we use a toy example in 2D and 3D.

A. Simulation experiments

In this section we provide the motivation for using ergodic control for exploration, and its significance for insertion tasks, using simulation of exploration behavior in 2D and 3D.

We use a GMM as the reference distribution in the space Ω with $L = 1$m. The GMM is chosen such that it has 6 equally weighted mixture components with its centers placed randomly in the exploration space Ω and each component is a spherical Gaussian with variance 0.01. As a first metric, we compute the average time taken to reach a spherical region with volume 0.5% of the volume of Ω. The spherical region is representative of the target detection region during exploration. For the insertion task, this corresponds to the set of end-effector states at which the peg is inside the hole. For all the trials, we fix a maximum duration of 1000s for detection (i.e., reaching the target region) and the magnitude of the point-mass-system velocity is constant and fixed to $u_{max} = 0.1m/s$. For the analysis we choose 10 different GMMs as described above and for each choice of GMM, we conduct 10 trials. For each trial, the center of the target region is chosen by sampling in Ω from the reference GMM and the point-mass system starts with the same initial state: $(0.5, 0.5)$ for 2D and $(0.5, 0.5, 0)$ for 3D.

We compare four different exploration strategies:

1) **Strategy 1:** *Ergodic exploration* (proposed approach),
2) **Strategy 2:** *Sampling-based exploration*
3) **Strategy 3:** *Cylindrical spiral*, as a representative of sweeping patterns,
4) **Strategy 4:** *Mixture of ellipsoidal spirals*, as a sweeping pattern customized for GMM.

Fig. 7 shows an example of exploration behaviors for these different strategies.

In strategy 2, we explore by tracking samples from the GMM sequentially.[19] For an arbitrary reference probability distribution, Markov Chain Monte Carlo approaches are conventionally used to obtain the samples. In this approach, the simulated system tracks GMM samples using a constant speed, with a new sample being computed every time the previous one is reached. Unlike ergodic exploration, such approaches based on sampling from the reference distribution are typically inefficient at handling distributed information [2].

In strategy 3, we use an Archimedean spiral for 2D and its cylindrical extension for 3D (see Fig. 7). These are conventionally used in robotics as heuristic search strategies for uniform coverage in 2D and 3D search spaces (i.e. for uniform reference probability distributions) [35] [36]. Strategy 4 is similar to strategy 3 but uses spherical/ellipsoidal spiral trajectories that are customized to sweep the GMM search space. In this approach, the Gaussians are swept in sequence with spherical/ellipsoidal spiral paths starting from the centers of the Gaussian and sweeping the area up to a given number of standard deviations before moving to the next component. These approaches suffer from the curse of dimensionality and perform poorly for $d > 3$. Moreover, they require careful tuning of hyperparameters to generate efficient spiral paths. Furthermore, the resulting trajectories are deterministic and do not consider the stochasticity of target detection. Namely, the trajectory generated by such sweeping pattern passes through a given point in the search space only once. If the measurement system fails to detect the target during its first pass, the strategy has no future possibilities for detection.

The first metric we use for comparison is the average time taken to reach the target region for the first time. Table III shows the obtained results for 2D and 3D. We see that, on average, ergodic exploration reaches the target region faster due to its multiscale search behavior. Additionally, ergodic control has a 100% success rate. Despite being slower, spiral search is equally successful, but this success comes at the cost of optimally choosing the spiral parameters. This is often cumbersome in practice, especially in higher dimensions and considering that the tolerance of the detection region is often not known with high certainty.

A desirable property for exploration strategies is that the system takes into account the already visited regions to cover the unvisited regions more often. In order to evaluate this property, we consider a second metric: the cumulative average time to reach the target region over several successful attempts. We define this as $\frac{T_c}{c}$, where c is the number of successful attempts and T_c is the cumulative time until successful attempt c. In this evaluation, as soon as the system reaches the boundary of the target region, we re-initialize it to the initial state and repeat this process for a fixed number of times. The results in Fig. 8 show that, for ergodic control, the cumulative average decreases with the number of successful attempts and converges to a fixed value. This suggests that the ergodic exploration tries to visit the unexplored regions

[19]We use the *numpy* Python package for sampling Gaussian distributions.

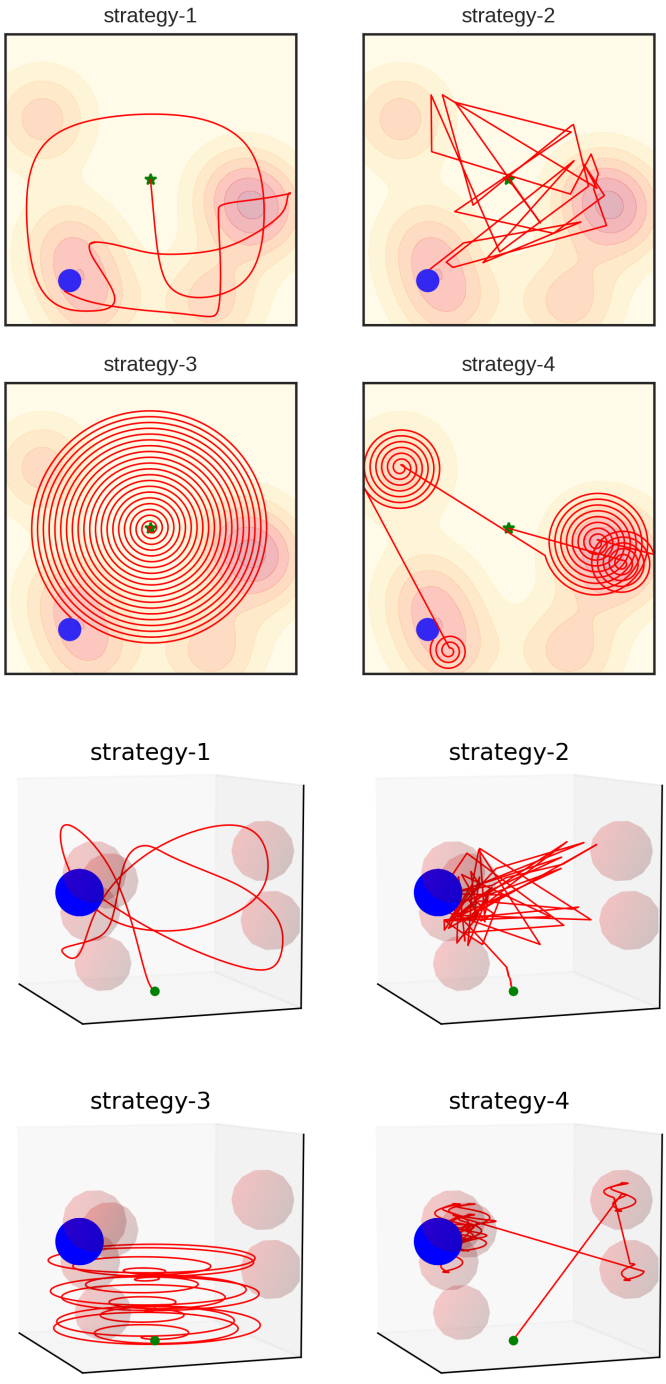


Fig. 7: Example trajectories (red) of the four different exploration strategies to reach an target region (blue sphere) within a reference probability distribution (in this case a GMM) for $d = 2$ (top) and $d = 3$ (bottom). The green point indicates the initial state of the point-mass system. The goal is to reach this target region the information about which is known to the search strategies only through the reference probability distribution.

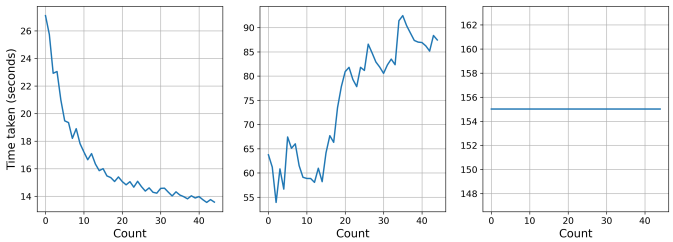


Fig. 8: Cumulative average time to reach a specified target region (spherical region with 0.5% of the volume of Ω) with ergodic exploration (**left**), sampling-based strategy (**center**) and spiral movement (**right**). The x-axis represents the number of attempts to reach the target and the y-axis represents the cumulative average of the time taken to reach the target at each number of attempts. Every time the target is reached, the system is re-initialized to the same initial state (starting a new attempt). The ergodic controller is aware of the lack of exploration inside the target region and tries to visit it more frequently. The cumulative average therefore converges to the time it takes to go from the initial state directly to the target at every re-initialization.

more frequently as the number of attempts increases. In this case, the value that the cumulative average converges to is the time it takes to go from the initial state directly to the target at every re-initialization. This is an essential feature for insertion tasks as the exploration inside the hole (representing successful insertion of the peg) is not easy due to obstacles (e.g., uncertainties and collision of the surface of the hole against the peg) and we need the exploration strategy to drive exploration towards the unexplored region as the time evolves. This is an inherent property of ergodic exploration that the other approaches lack. This property plays a crucial role to cope with the stochasticity involved either in the measurement systems for detection, or the dynamics of the process (e.g., insertion task). To exploit this feature in real robot experiments, it is necessary that the ergodic controller is implemented *online* on the robot manipulator, i.e., that the controller knows the actual observed end-effector states. Our proposed algorithm for ergodic controller allows this online implementation on robot manipulators for $d = 6$, which is demonstrated in the next section.

In the insertion task, the target region for detection corresponds to a set of states of the peg that are necessary to be passed through for a successful insertion of the peg. This information is obtained from the reference probability distribution for exploration in the search space. Considering stochasticity is important for a search strategy to be useful in practice. In general, stochasticity may arise either from the measurement system (e.g., uncertainties in the location of the hole, grasp of the peg or manufacturing defects) and/or the dynamics of the system (e.g., stochasticity in the contact dynamics involved in the insertion). For example, during the insertion, the peg might be at the correct relative location to the hole according to the sensor system, but the insertion may still not be successful every time in that configuration due to the stochasticity of the process. We need the search strategy to explore more often in these target regions (correct configurations for insertion) to increase the likelihood of insertion. Ergodic control considers this stochasticity by driving the system to regions in the state space such that the amount of time it spends there is in proportion to the probability mass of that region. For more

TABLE III: Average time taken to reach a target region for different exploration strategies.

	Strategy	Success rate		Time taken (seconds)
		# Trials	# Success	
2D	Strategy 1	100	100	**66.9**
	Strategy 2	100	96	106.8
	Strategy-3	100	100	122.9
	Strategy-4	100	98	155.4
3D	Strategy-1	100	100	**84.7**
	Strategy-2	100	92	141.4
	Strategy-3	100	98	292.3
	Strategy-4	100	95	247.5

details on search strategies and their desired characteristics, see the seminal work of Koopman on the theory of search [37], [38], [39]. Ergodic exploration satisfies the standards for optimal search behavior set by Koopman. Although strategy 3 and 4 do not satisfy these properties, we included them in our evaluation for completeness.

B. Experimental Setup

We use a torque-controlled 7-axis Franka Emika Panda robot, with an insertion task based on the Siemens gear set benchmark (see Fig. 9)[20], by using the 25.4mm-diameter peg and the 26.29mm-diameter receptacle, with the length of insertion of 47mm. We employed a Cartesian impedance controller to control the robot end-effector by computing a desired Cartesian wrench according to

$$\hat{\boldsymbol{f}} = \boldsymbol{K}_p \begin{bmatrix} \hat{\boldsymbol{p}} - \boldsymbol{p} \\ \mathrm{Log}\left(\hat{\boldsymbol{q}} * \bar{\boldsymbol{q}}\right) \end{bmatrix} - \boldsymbol{K}_d \begin{bmatrix} \dot{\boldsymbol{p}} \\ \boldsymbol{\omega} \end{bmatrix},$$

where $\hat{\boldsymbol{p}} \in \mathbb{R}^3$, $\hat{\boldsymbol{q}} \in \mathcal{S}^3$ are, respectively, the desired position and orientation of the end-effector (with $\hat{\boldsymbol{q}}$ obtained from (14)), $\boldsymbol{p}, \boldsymbol{q}, \dot{\boldsymbol{p}}, \boldsymbol{w}$ are the end-effector position, orientation, linear and angular velocity and $\boldsymbol{K}_p, \boldsymbol{K}_d$ are 6×6 diagonal stiffness and damping gains, experimentally set to $\boldsymbol{K}_p = \mathrm{diag}\left(500, 500, 500, 160, 160, 160\right)$ and $\boldsymbol{K}_v = \mathrm{diag}\left(40, 40, 40, 10, 10, 10\right)$. The symbol $*$ denotes the unit quaternion product and $\bar{\boldsymbol{q}}$ the conjugate of quaternion $\boldsymbol{q}$. $\mathrm{Log}(\cdot)$ is the logarithmic map defined in (9).

We obtain the desired robot joint torques with $\hat{\boldsymbol{\tau}} = \boldsymbol{J}(\boldsymbol{\theta})^\top \hat{\boldsymbol{f}} + \boldsymbol{g}\left(\boldsymbol{\theta}\right) + \boldsymbol{h}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})$, where $\boldsymbol{\theta}, \dot{\boldsymbol{\theta}} \in \mathbb{R}^7$ are the robot joint positions and velocities and $\boldsymbol{J} \in \boldsymbol{R}^{6\times7}$, $\boldsymbol{g} \in \mathbb{R}^7$, $\boldsymbol{h} \in \mathbb{R}^7$ are the Jacobian matrix of the end-effector, gravity and Coriolis terms. Note that the impedance gains were selected such that the robot remained compliant enough to safely interact with the environment during exploration, while still being able to track the ergodic trajectory.

We compare three different implementations of our approach. First, a *closed loop* version where the ergodic controller runs in real-time on the robot as an online coverage problem. In that case, at every time step, we read the end-effector pose and feed it back to the ergodic controller, which computes the next command based on the current state. Second, an *open loop* version where the controller tracks a

[20]https://new.siemens.com/us/en/company/fairs-events/robot-learning.html

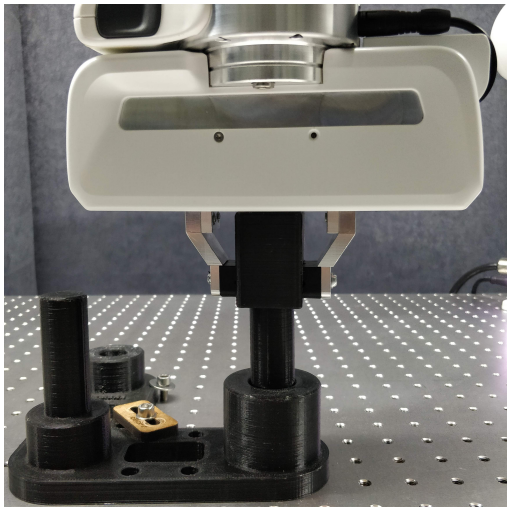
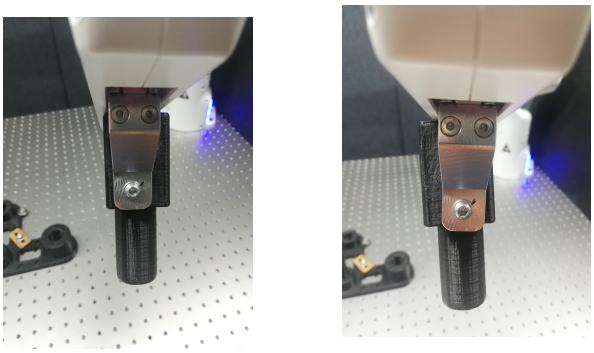


Fig. 9: The Siemens gear benchmark used for evaluation of the peg-in-hole insertion task using ergodic control.



(a) A grasp with an offset in position

(b) A grasp with an offset in orientation

Fig. 10: Two instances of grasps typically appearing when testing the ergodic control for insertion. The demonstrations included different types of grasps to let the robot cope with this uncertainty during ergodic exploration.

reference ergodic trajectory computed offline (as used in [21] for a lower dimensional state space). Lastly, GMM-sampling-based exploration as presented in Section VI-A.

C. Ergodic Controller Initialization

In our setup, the location of the hole is fixed (but unknown to the robot) and the main source of uncertainty comes from the different possible grasps of the peg by the end-effector, see Fig. 10. We model these uncertainties using a probability density function that indicates the importance of spending time in each region of the state space of the robot end-effector. We use ergodic control to insert the peg under such uncertainties.

To model the reference probability distribution for ergodic exploration of insertion, we collected $M = 204$ data points using kinesthetic teaching, which corresponds to less than 2 minutes of recording, see Fig. 11. Each datapoint corresponds to the position and orientation of the end-effector holding the peg. The datapoints in the vicinity of the location of the hole were recorded for successful insertions with different orientation and position offsets inherent to the grasps of the peg. To give higher importance to insertion, almost half the

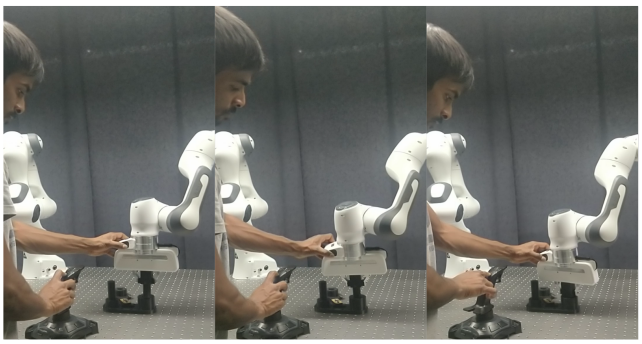


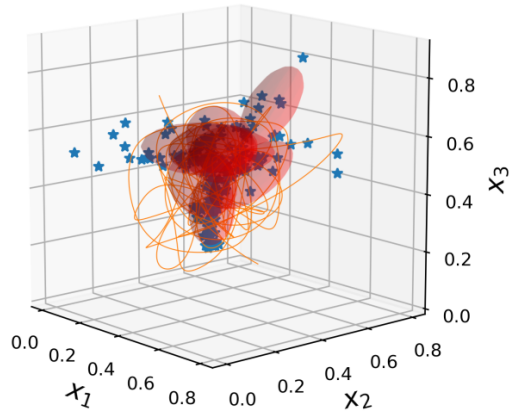
Fig. 11: Human demonstration of peg-in-hole insertion task. Datapoints are collected for different grasps of the peg through kinesthetic teaching to show the regions in the vicinity of the receptacle to be used by the ergodic controller.

data points were taken from the states corresponding to the peg within the hole. A variation of about 15mm in position of the grasp for each axis and about $\sim 10^\circ$ variation in the orientation of the grasp were demonstrated (see accompanying video of the experiment).

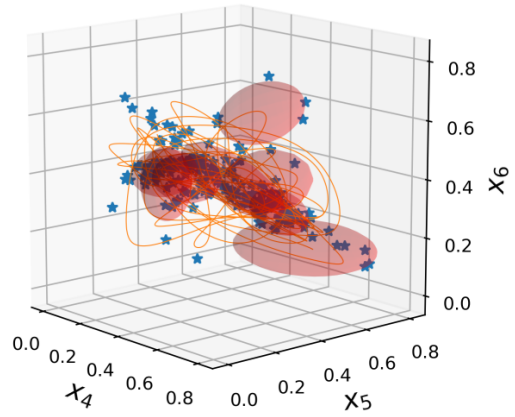
Once data are collected, we preprocess the end-effector orientations as described in Section IV-B. Subsequently, we concatenate the position data with the obtained 3D orientation representation into a 6D vector. We then transform the data into the domain of ergodic control Ω with $L = 1$ (ergodic space) using a bijective linear transformation. We model the data points in this transformed space using a Gaussian mixture model (GMM) with full covariances. We empirically selected 8 mixture components with a minimal isotropic covariance prior of 5×10^{-3}. Figure 12 shows the obtained GMM for position and orientation (marginal distributions). The GMM is used as reference probability distribution $P(\boldsymbol{x})$ for ergodic control in Ω. With $K = 10$ and $N = 10$, the computation of Fourier coefficients $\hat{\boldsymbol{\mathcal{W}}}$ for the reference probability distribution took less than 2 minutes. The closed-loop ergodic controller could be run at 100Hz ($dt = 0.01$ in Algorithm 2) on the robot. The same settings are applied to the open-loop version of the insertion task. In this case, the trajectory is generated offline using ergodic control. It is then tracked using the above-mentioned settings of the impedance control. Figure 12 shows examples of generated ergodic trajectories. We allow a maximum of 40 seconds for insertion. Figure 13 shows trajectory generated from ergodic control in one of the trials of the experiment.

D. Experimental Results

The obtained results, summarized in Table IV, show that the *closed loop* approach clearly outperforms the other approaches that do not consider the history of observed states during the exploration. Indeed, while using the former, the robot was able to successfully insert the peg in the hole in 20 out of the 20 trials, where 10 were performed using the grasp in Fig. 10a and the other 10 using the one in Fig. 10b. On average, the insertion using the *closed loop* approach succeeded in $9.9s$ with a standard deviation of $8.5s$. Figure 14 shows snapshots of the insertion using ergodic exploration in the *closed loop* setting. Note that the only required input was a



(a) Position distribution



(b) Orientation distribution

Fig. 12: Position and orientation marginals from the distribution used for full pose exploration (the figure does not show correlations between position and orientation). The 6D pose distribution is encoded as a GMM with 8 components (red ellipsoids) and full covariances, trained on a dataset of $M = 204$ datapoints (blue points). The trajectory for the exploration (orange lines) is generated by ergodic control.

set of demonstrations to show the robot the regions it should explore. This has proven important to deal with the uncertainty in the peg grasping pose. Note that interaction forces during exploration can also cause the grasp pose to change due to the limited gripping force of the robot (either from hardware and software limitations, or set on purpose to handle fragile objects). Our experimental results show that the closed loop ergodic strategy is also robust in these situations.

The open loop version succeeded in only 2 out of 10 trials and the naive approach of GMM-sampling-based exploration succeeded in none of the 5 trials. This shows the importance of exploiting the history of the real observed end-effector poses to compute control commands. This is particularly important for tasks requiring contacts with the environment, where the commands need to be re-evaluated according to the history of poses retrieved by the impedance controller. Notably, this allows the use of low gains to remain compliant and enable safe contacts. In the closed loop approach, the algorithm is aware of the locations that were previously effectively visited, which is exploited to fulfill the insertion task in an online and robust fashion.[21]

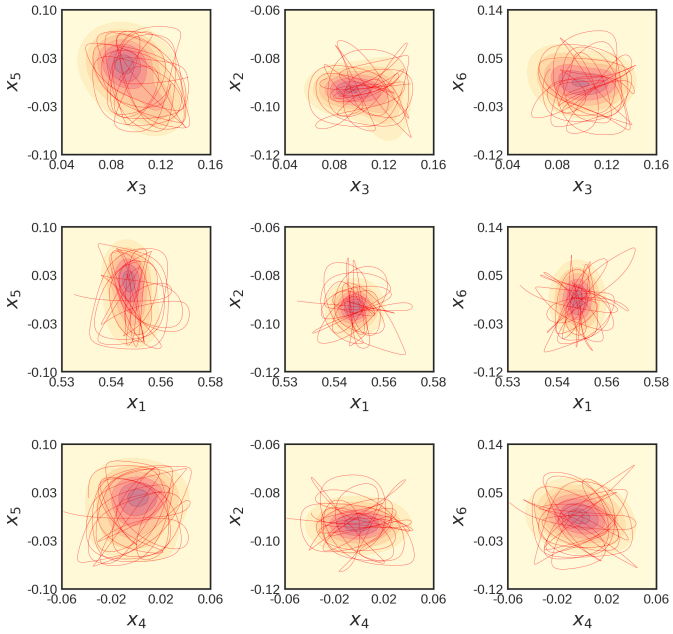


Fig. 13: The robot trajectory generated by ergodic controller offline for the insertion task.

TABLE IV: Performance of the peg-in-hole task for different variations of the grasps.

Strategy	**Success rate**		**Time taken**
	# Trials	Successful Trials	**(seconds)**
Closed loop	20	20	9.9 ± 8.5
Open loop	10	2	-
GMM-sampling	5	0	-

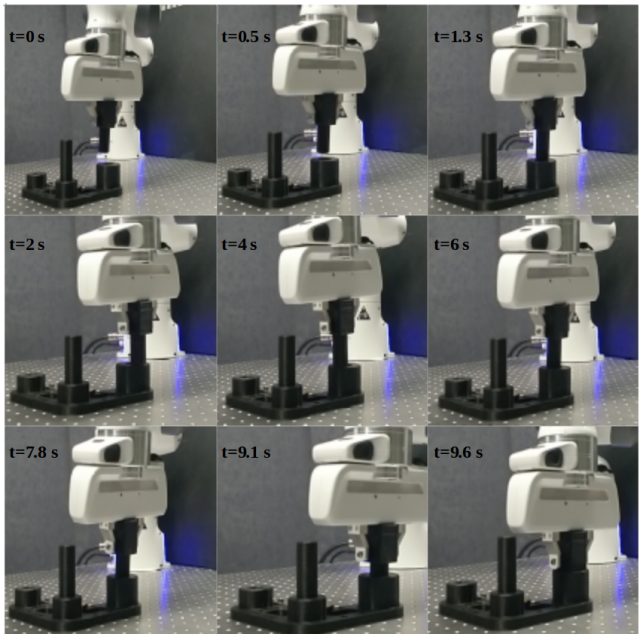


Fig. 14: Snapshots of an insertion using ergodic control in closed loop.

[21] A video of the experiment is available at https://sites.google.com/view/ergodic-exploration/.

VII. Conclusion and Further Work

We proposed a solution to the multidimensional ergodic exploration problem with $d > 3$, which was previously considered to be intractable for applications in robotic manipulation. The proposed approach relies on tensor train decompositions and is evaluated in simulated examples of target detection and a real robot experiment using a peg-in-hole insertion task. The obtained results show that ergodic control has a 100% success rate in all tasks, succeeding faster than the competing approaches. In addition to the novelty of exploiting ergodic control in an online fashion for insertion tasks, the computational techniques we used demonstrate how algorithms in robotics, in general, can benefit from tensor methods to overcome the limitations in computational speed and storage requirements.

The main challenges of extending ergodic control to state space of more than 3 dimensions concern the computation of Fourier coefficients (preprocessing step) and the speed of the control loop (for online implementation). This article addressed both of these challenges using the properties of tensor train decomposition. We leveraged this improved ergodic control formulation to propose a sensorless strategy for peg-in-hole insertion tasks. We validated our approach with a compliant robot manipulator, where the 6D regions to explore are obtained from kinesthetic human demonstrations. Our experimental results show that by using our approach, the robot is capable of achieving challenging insertion tasks without force/torque sensors.

By using the reproducible Siemens gears benchmark, insertion tasks with unknown gripping variations could be achieved in less than 10 seconds on average. This is, in part, due to the multi-scale exploration behavior that is inherent to the ergodic metric we employed. With such metric, the resulting controller first crudely explores the region of interest, and then progressively refines the search until insertion, efficiently exploiting information about the already covered regions. This is also due to the fast processing that we propose (through tensor train factorization), that allows ergodic control to be run in an online manner, even by considering distributions of full end-effector poses. Indeed, re-estimating the control commands on-the-fly allows the proposed ergodic control strategy to be used together with an impedance controller with low gains, which is important for tasks requiring contacts with the environment.

There is still room for improvement of the proposed approach. First, designing demonstrations for insertion tasks can be further optimized. In future work, we plan to study alternative demonstration strategies (including different distributions), which could speed up the insertion time using ergodic control. We will also apply the insertion tasks to different physical setups by using different sensory modalities. We also plan to study applications of our algorithm to other applications in robotics, either requiring exploration, active sensing, or, more generally, for applications requiring multidimensional basis functions with a low rank structure.

Appendix A
Proof of Fourier Coefficients Decomposition

A one-dimensional integral

$$s = \int_{x=0}^{l} f(x)dx$$

can be computed numerically with the Gaussian-Quadrature (GQ) rule as

$$s = \sum_{j=1}^{N} \alpha_j f(x_j),$$

where N represents the degree of approximation (specified by the user), x_j represents the discretization points, and α_j are the corresponding weights. For any polynomial function of degree less than $2N-1$, the above summation gives exact result without any error in the integration. For a given N, x_j and α_j can be computed and are readily available using software packages for scientific computing.

Now recall that in (1), we want to evaluate

$$\hat{\boldsymbol{\mathcal{W}}}_{\boldsymbol{k}} = \int_{x_1=0}^{L} \cdots \int_{x_d=0}^{L} P(\boldsymbol{x}) \boldsymbol{\Phi}_{\boldsymbol{k}}(\boldsymbol{x}) dx_1 \cdots dx_d, \ \forall \boldsymbol{k}. \quad (15)$$

Let x_j be the discretization points and α_j the corresponding weights, with $j \in \{1, \ldots, N\}$, obtained from the GQ rule for a given N. Let $\mathcal{J} = \{\boldsymbol{j} = (j_1, \ldots, j_d) : j_i \in \{1, \ldots, N\}\}$ be the index set. We can discretize the domain Ω at $\boldsymbol{x}_{\boldsymbol{j}} = (x_{j_1}, \ldots, x_{j_d})$, with $\boldsymbol{j} \in \mathcal{J}$. Let $\boldsymbol{\mathcal{P}}$ be the tensor formed by evaluating the reference distribution $P(\boldsymbol{x})$ at the discretization points, i.e., $\boldsymbol{\mathcal{P}}_{\boldsymbol{j}} = P(\boldsymbol{x}_{\boldsymbol{j}})$.

Then, we can evaluate the above integral using GQ as

$$\begin{aligned} \hat{\boldsymbol{\mathcal{W}}}_{\boldsymbol{k}} &= \sum_{\boldsymbol{j} \in \mathcal{J}} \alpha_{j_1} \cdots \alpha_{j_d} P(\boldsymbol{x}_{\boldsymbol{j}}) \boldsymbol{\Phi}_{\boldsymbol{k}}(\boldsymbol{x}_{\boldsymbol{j}}) \\ &= \sum_{\boldsymbol{j} \in \mathcal{J}} \alpha_{j_1} \cdots \alpha_{j_d} P(\boldsymbol{x}_{\boldsymbol{j}}) \phi_{i_1}(x_{j_1}) \cdots \phi_{i_d}(x_{j_d}), \ \forall \boldsymbol{k} \in \mathcal{K}. \end{aligned} \quad (16)$$

Discretizing $P(\boldsymbol{x})$ at the GQ points $\boldsymbol{x}_{\boldsymbol{j}} = (x_{j_1}, \ldots, x_{j_d})$ with $\boldsymbol{j} \in \mathcal{J}$, we can get the tensor $\boldsymbol{\mathcal{P}}$, with $\boldsymbol{\mathcal{P}}_{\boldsymbol{j}} = P(\boldsymbol{x}_{\boldsymbol{j}})$.

Consider a TT-representation of $\boldsymbol{\mathcal{P}}$ given by the TT-cores $(\boldsymbol{\mathcal{P}}^1, \boldsymbol{\mathcal{P}}^2, \ldots, \boldsymbol{\mathcal{P}}^d)$, so that for $\boldsymbol{j} = (j_1, \ldots, j_d) \in \mathcal{J}$ we have

$$\boldsymbol{\mathcal{P}}_{\boldsymbol{j}} = \boldsymbol{\mathcal{P}}^1_{:,:,j_1} \boldsymbol{\mathcal{P}}^2_{:,:,j_2} \cdots \boldsymbol{\mathcal{P}}^d_{:,:,j_d}.$$

Substituting the above expression in (16) yields

$$\begin{aligned} \hat{\boldsymbol{\mathcal{W}}}_{\boldsymbol{k}} &= \sum_{\boldsymbol{j} \in \mathcal{J}} \alpha_{j_1} \cdots \alpha_{j_d} \boldsymbol{\mathcal{P}}^1_{:,:,j_1} \boldsymbol{\mathcal{P}}^2_{:,:,j_2} \cdots \\ &\qquad \cdots \boldsymbol{\mathcal{P}}^d_{:,:,j_d} \phi_{k_1}(x_{j_1}) \cdots \phi_{k_d}(x_{j_d}) \\ &= \Big(\sum_{j_1=1}^{N} \alpha_{j_1} \boldsymbol{\mathcal{P}}^1_{:,:,j_1} \phi_{k_1}(x_{j_1}) \Big) \cdots \Big(\sum_{j_d=1}^{N} \alpha_{j_d} \boldsymbol{\mathcal{P}}^d_{:,:,j_d} \phi_{k_d}(x_{j_d}) \Big). \end{aligned} \quad (17)$$

Also, we know that the TT-decomposition of $\hat{\boldsymbol{\mathcal{W}}}$ takes the form

$$\hat{\boldsymbol{\mathcal{W}}}_{\boldsymbol{k}} = \hat{\boldsymbol{\mathcal{W}}}^1_{:,:,k_1} \cdots \hat{\boldsymbol{\mathcal{W}}}^d_{:,:,k_d}, \ \forall \boldsymbol{k} \in \mathcal{K}. \quad (18)$$

Comparing (17) with (18), we obtain an expression for the TT-cores of the TT-decomposition of $\hat{\boldsymbol{\mathcal{W}}}$ as

$$\hat{\boldsymbol{\mathcal{W}}}^{i}_{:,:,k} = \sum_{j=1}^{N} \alpha_j \boldsymbol{\mathcal{P}}^{i}_{:,:,j} \phi_k(x_j), \quad \begin{matrix} \forall k \in (1,\dots,K), \\ \forall i \in (1,\dots,d). \end{matrix} \tag{19}$$

REFERENCES

[1] A. Hubenko, V. A. Fonoberov, G. Mathew, and I. Mezic, "Multiscale adaptive search," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 41, no. 4, pp. 1076–1087, 2011.

[2] L. M. Miller, Y. Silverman, M. A. MacIver, and T. D. Murphey, "Ergodic exploration of distributed information," *IEEE Transactions on Robotics*, vol. 32, pp. 36–52, 2016.

[3] G. Mathew and I. Mezić, "Metrics for ergodicity and design of ergodic dynamics for multi-agent systems," *Physica D: Nonlinear Phenomena*, vol. 240, no. 4-5, pp. 432–442, 2011.

[4] G. Mathew, S. Kannan, A. Surana, S. Bajekal, and K. R. Chevva, "Experimental implementation of spectral multiscale coverage and search algorithms for autonomous UAVs," in *AIAA Guidance, Navigation, and Control (GNC) Conference*, 2013, p. 5182.

[5] A. Mavrommati, E. Tzorakoleftherakis, I. Abraham, and T. D. Murphey, "Real-time area coverage and target localization using receding-horizon ergodic exploration," *IEEE Transactions on Robotics*, vol. 34, pp. 62–80, 2018.

[6] S. Calinon, "Mixture models for the analysis, edition, and synthesis of continuous time series," in *Mixture Models and Applications*, N. Bouguila and W. Fan, Eds. Springer, 2019, pp. 39–57.

[7] L. M. Miller and T. D. Murphey, "Trajectory optimization for continuous ergodic exploration," in *American Control Conference*. IEEE, 2013, pp. 4196–4201.

[8] L. Dressel and M. J. Kochenderfer, "Tutorial on the generation of ergodic trajectories with projection-based gradient descent," *IET Cyper-Phys. Syst.: Theory & Appl.*, vol. 4, pp. 89–100, 2019.

[9] E. Ayvali, H. Salman, and H. Choset, "Ergodic coverage in constrained environments using stochastic trajectory optimization," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 5204–5210.

[10] I. Abraham, A. Prabhakar, and T. Murphey, "An ergodic measure for active learning from equilibrium," *ArXiv*, vol. abs/2006.03552, 2020.

[11] T. Sahai, G. Mathew, and A. Surana, "A chaotic dynamical system that paints and samples," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 10 760 – 10 765, 2017, 20th IFAC World Congress.

[12] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM review*, vol. 51, no. 3, pp. 455–500, 2009.

[13] A. Cichocki, D. P. Mandic, L. D. Lathauwer, G. Zhou, Q. Zhao, C. F. Caiafa, and A. H. Phan, "Tensor decompositions for signal processing applications: From two-way to multiway component analysis," *IEEE Signal Processing Magazine*, vol. 32, pp. 145–163, 2015.

[14] S. Rabanser, O. Shchur, and S. Günnemann, "Introduction to tensor decompositions and their applications in machine learning," *ArXiv*, vol. 1711.10781, 2017.

[15] M. B. Horowitz, A. Damle, and J. W. Burdick, "Linear Hamilton Jacobi Bellman equations in high dimensions," *53rd IEEE Conference on Decision and Control*, pp. 5880–5887, 2014.

[16] A. A. Gorodetsky, S. Karaman, and Y. M. Marzouk, "Efficient high-dimensional stochastic optimal motion control using tensor-train decomposition," in *Robotics: Science and Systems*, 2015.

[17] S.-k. Yun, "Compliant manipulation for peg-in-hole: Is passive compliance a key to learn contact motion?" in *IEEE International Conference on Robotics and Automation*. IEEE, 2008, pp. 1647–1652.

[18] P. R. Giordano, A. Stemmer, K. Arbter, and A. Albu-Schaffer, "Robotic assembly of complex planar parts: An experimental evaluation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2008, pp. 3775–3782.

[19] H. Park, J. Park, D.-H. Lee, J.-H. Park, M.-H. Baeg, and J.-H. Bae, "Compliance-based robotic peg-in-hole assembly strategy without force feedback," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 8, pp. 6299–6309, 2017.

[20] M. P. Polverini, A. M. Zanchettin, S. Castello, and P. Rocco, "Sensorless and constraint based peg-in-hole task execution with a dual-arm robot," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 415–420.

[21] D. Ehlers, M. Suomalainen, J. Lundell, and V. Kyrki, "Imitating human search strategies for assembly," in *International Conference on Robotics and Automation (ICRA)*, Montréal, Canada, May 2019, pp. 7821–7827.

[22] M. J. A. Zeestraten, I. Havoutis, J. Silvério, S. Calinon, and D. G. Caldwell, "An approach for imitation learning on Riemannian manifolds," *IEEE Robotics and Automation Letters (RA-L)*, vol. 2, no. 3, pp. 1240–1247, June 2017.

[23] S. Calinon, "Gaussians on Riemannian manifolds: Applications for robot learning and adaptive control," *IEEE Robotics and Automation Magazine (RAM)*, vol. 27, no. 2, pp. 33–45, June 2020.

[24] P. M. Kroonenberg, *Applied multiway data analysis*. John Wiley & Sons, 2008, vol. 702.

[25] G. Mathew, A. Surana, and I. Mezic, "Uniform coverage control of mobile sensor networks for dynamic target detection," *49th IEEE Conference on Decision and Control (CDC)*, pp. 7292–7299, 2010.

[26] L. Grasedyck, D. Kressner, and C. Tobler, "A literature survey of low-rank tensor approximation techniques," *GAMM-Mitteilungen*, vol. 36(1), pp. 53–78, 2013.

[27] I. V. Oseledets, "Tensor-train decomposition," *SIAM J. Scientific Computing*, vol. 33, pp. 2295–2317, 2011.

[28] N. Lee and A. Cichocki, "Fundamental tensor operations for large-scale data analysis using tensor network formats," *Multidimensional Systems and Signal Processing*, vol. 29, no. 3, pp. 921–960, 2018.

[29] D. V. Savostyanov and I. V. Oseledets, "Fast adaptive interpolation of multi-dimensional arrays in tensor train format," *The 2011 International Workshop on Multidimensional (nD) Systems*, pp. 1–8, 2011.

[30] I. Oseledets and E. Tyrtyshnikov, "TT-cross approximation for multidimensional arrays," *Linear Algebra and its Applications*, vol. 432, no. 1, pp. 70–88, 2010.

[31] N. Kishore Kumar and J. Schneider, "Literature survey on low rank approximation of matrices," *Linear and Multilinear Algebra*, vol. 65, no. 11, pp. 2212–2244, 2017.

[32] C. Lubich, I. V. Oseledets, and B. Vandereycken, "Time integration of tensor trains," *SIAM J. Numerical Analysis*, vol. 53, pp. 917–941, 2015.

[33] D. Bigoni, A. P. Engsig-Karup, and Y. M. Marzouk, "Spectral tensor-train decomposition," *SIAM Journal on Scientific Computing*, vol. 38, no. 4, pp. A2405–A2439, 2016.

[34] S. Dolgov and D. Savostyanov, "Parallel cross interpolation for high-precision calculation of high-dimensional integrals," *Computer Physics Communications*, vol. 246, p. 106869, 2020.

[35] Hyeonjun Park, Ji-Hun Bae, Jae-Han Park, Moon-Hong Baeg, and Jaeheung Park, "Intuitive peg-in-hole assembly strategy with a compliant manipulator," in *IEEE ISR*, 2013, pp. 1–5.

[36] J. C. Triyonoputro, W. Wan, and K. Harada, "Quickly inserting pegs into uncertain holes using multi-view images and deep network trained on synthetic data," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 5792–5799.

[37] B. O. Koopman, "The theory of search. I. kinematic bases," *Operations research*, vol. 4, no. 3, pp. 324–346, 1956.

[38] ——, "The theory of search. II. target detection," *Operations research*, vol. 4, no. 5, pp. 503–531, 1956.

[39] ——, "The theory of search: III. the optimum distribution of searching effort," *Operations research*, vol. 5, no. 5, pp. 613–626, 1957.