Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnline CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 56-2 (2023) 306–313

Active learning of time-optimal motion for an industrial manipulator with variable payload

Jiayun Li * **Arne Wahrburg** ** **Nima Enayati** **

* *Department of Mechanical Engineering and Transport Systems, Technical University of Berlin, Berlin, Germany. (e-mail: jiayun.li@campus.tu-berlin.de)*

** *ABB Corporate Research, Ladenburg, Germany. (e-mail: {arne.wahrburg, nima.enayati}@de.abb.com)*

Abstract: Generating optimal motion for robotic systems remains an active field of research, despite its relatively long history. In addition to being optimal with respect to a defined cost, such a motion must respect the electro-mechanical limitations of the system to be of use in real-world applications. Furthermore, in an industrial application where offline planning is not possible due to a priori unknown targets and payloads, the available time budget for computations is often highly limited, making many optimal motion planning approaches unsuitable. This work introduces a supervised learning-based motion planner for industrial manipulators that closely retains the superior performance of an optimal motion planner while requiring a fraction of the computations at runtime. The proposed motion planner implicitly considers the dynamics of the robot and therefore satisfies not only kinematic limits but also those of the actuators. The proposed method is compared to state-of-the-art approaches and demonstrated on a real robot.

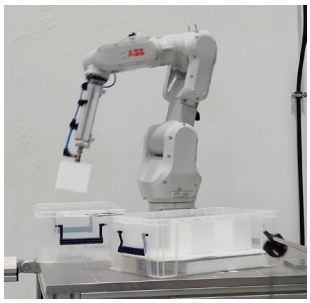
Keywords: Robots manipulators, Data-driven optimal control, Nonparametric methods

1. INTRODUCTION

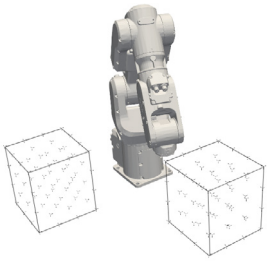
Robot motion planning is among the core components of any robotic system. A robot motion, defined by a geometrical path and the timing along the path, must satisfy several requirements to reliably accomplish the tasks defined by the user of such a system. The users of industrial manipulators are often interested in the robot accurately and predictably following a programmed path, and the speed at which the robot can conduct such a motion is of interest, too. Faster robots are key to higher productivity. Therefore achieving time-optimality has been one of the longest-running topics of robotic research for decades Laumond et al. (1998). Optimality can only be achieved if both the path and the timing are determined such that the robot operates at its electro-mechanical limits. Introductory textbook approaches such as trapezoidal velocity profile Siliciano et al. (2010) are valuable resources for education purposes or applications that do not require high performance. However, in general generating a trajectory merely based on kinematic limits such as joint position, speed, and acceleration will likely lead to under- or over-utilization of actuators, in turn resulting in sub-optimal performance or control/hardware issues while attempting to track an unfeasible trajectory. Furthermore, it is often not possible to devise limits for acceleration or jerk, as these quantities often do not correspond to any physical limitation of the robot mechanics or its actuators. Considering the actuation limits of a manipulator, however, means including the dynamics of the manipulator and its payloads. The actuation limits themselves can in many cases be dependent on position or speed adding further complexity to the motion planning problem. Therefore, such a planning problem is often solved through a local optimization that can include hundreds of computationally expensive constraints for a typical 6 Degree-of-Freedom (DoF) robot, which may take a considerable amount of time. One can calculate such optimized trajectories offline and execute them at runtime. This approach will not be tangible if the manipulator is part of an application with targets and payloads that are unknown a priori. It can therefore be of interest to approximate such an optimized trajectory through machine learning to achieve a planner that potentially sacrifices some performance to gain an advantage in online computation time. The problem addressed by this work, however, is to generate time-optimal motion for given initial and final states of a manipulator within a realistic industrial application with limited computation time-budget and variable payload such as pick and place in Fig. 1.

A large number of works in the literature on machine learning for motion planning regard sensor-based planning of path or trajectory towards performing a task with an acceptable success rate, and ideally generalize to unseen scenarios (e.g., Qureshi et al. (2019)). We are more interested in the optimality and dynamic feasibility of the trajectories than generalizing beyond the training regions of the workspace. Although there are a small number of works, discussed in the related works section, that attempt similar problems to the one described here, to our knowledge, this work is unique in achieving a motion planner for a 6-DoF manipulator with a variable payload that is

10.1016/j.ifacol.2023.10.1585



(a) An ABB IRB1100-4-058 performing pick-and-place



(b) Sampled subsets of the robot workspace

Fig. 1. The application setup and workspace of interest

highly time-optimal and at the same time respects torque limits of the actuators. The novelties are further detailed in the next section.

2. RELATED WORK AND CONTRIBUTION

In data-based motion planning, the choice of trajectory parameterization method is of great importance. An ideal encoding captures the trajectory accurately in the smallest possible number of parameters. Dynamic Movement Primitives (DMPs) have become the de facto choice in reinforcement/imitation learning due to their elegant, stable, and efficient formulation Saveriano et al. (2021). As we are interested in encoding highly dynamic trajectories that do not violate joint torque limits, and since torques depend not only on position but also on velocity and acceleration, the trajectory parameterization method needs to retain accurately all the 3 quantities. However, standard DMPs produce abrupt changes in acceleration, which can also result in an instantaneous change in torque. Therefore, the standard DMPs can only represent a smooth trajectory at velocity-level. This has not been an issue in all the related works we have found that use DMPs (Forte et al. (2012), Calinon (2016), Fanger et al. (2016), Deniša et al. (2015), Weitschat et al. (2013), Pervez and Lee (2018)), as these works do not consider the dynamics of the manipulator or its actuation limits. This work proposes 3rd order DMPs as the trajectory parameterization method, that is found to be conducive to dynamically feasible trajectory generation.

Trajectory generation from data is in essence a regression problem, for which numerous methods have been proposed in the literature. Non-parametric methods are particularly widely used in robotics as they are flexible and explainable. Gaussian Process Regression (GPR) used in Forte et al. (2012) and Fanger et al. (2016) is a suitable approach that unlike the linear alternative, Locally Weighted Projection Regression (LWPR) Vijayakumar and Schaal (2000) does not require manually adjusted meta-parameters and local models. Lembono et al. (2020) compared k-Nearest Neighbor, GPR, and Bayesian Gaussian Mixture Regression generating trajectories for warm-starting an optimization based planner. They found GPR to have the highest success rate in generating kinematically feasible point-to-point motion. In Deniša et al. (2015), GPR was used to learn compliant movement primitives, which is a combination of the standard DMP and the torque profile of a specific task demonstration to achieve compliant motion and high precision trajectory tracking. These works, however, do not take the dynamics of the manipulator into account and cannot guarantee dynamic feasibility. In Weitschat et al. (2013), a local average method based on the Euclidean distance of boundary conditions was adopted to train a planner that can generate energy-optimal point-to-point and ball throwing motions for a 2-DoF robot. It is stated that some dynamic optimality is achieved. However, whether the generated trajectories are within the actual limits of the actuators is not discussed. In Pervez and Lee (2018), Gaussian mixture models were utilized to generate DMP parameters for learning tasks such as pushing an object to a target location. Objective of the work was to extrapolate to new regions of the workspace, the success criteria depended merely on position, and neither optimality nor feasibility of trajectories are investigated. Training a point-to-point motion planner for the entire workspace of the robot can require a prohibitive amount of data. One might resort to sampling only sections of the workspace related to the task of interest to reduce the needed data. Nevertheless, when planning for several DoFs, adopting an efficient sampling strategy is fundamental. This is sometimes referred to as active learning. In Maeda et al. (2017), the variance measure of the GPR was used to evaluate whether the planner needs a new demonstration at runtime. The work was intended for a motion that reaches an arbitrary final state from a fixed start state without task space constraints, which has limited industrial application. Furthermore, as discussed in the methods section, active sampling based on variance can produce too many samples in the space boundaries and therefore can be less efficient compared to the mutual information measure used in this work Krause et al. (2008). Active learning has several other applications in the generalization of DMPs, e.g., Saveriano and Lee (2018), Conkey and Hermans (2019), Girgin et al. (2020), Kulak et al. (2021), however, where demonstrations are actively sampled to achieve a specific manipulation task. Such approaches are not suitable for predicting the trajectory given deterministic boundary conditions, as is our interest. In the application of sparse GPR in DMP learning we found no relevant literature. The contribution of this work includes the combination of the trajectory encoding method and the efficient learning algorithm, namely:

1. Continuous acceleration trajectory encoding with 3rd order DMP[3], which is critical for time-optimal motion within torque constraints.
2. The use of sparse Gaussian process regression to improve the scalability of the learning algorithm to address the complexity posed by variable robot payload and combinatorial boundary conditions.
3. The application of mutual information-based active learning strategy to efficiently sample in a constrained Cartesian task space.

3. METHODS

The main problem addressed in this work is to generate a time-optimal point-to-point trajectory for a d DoF industrial robot in a short amount of time (tens of milliseconds) given a query consisting of an initial joint position $\mathbf{q}_i$ [1], a

[1] In this paper, we present scalars in normal type, vectors in bold lowercase, and matrices in bold uppercase. Any additional symbols used will be explained upon their first appearance.

final joint position $\mathbf{q}_f$, and the payload μ carried by the robot. The joints' velocity and acceleration at the initial and final states are zero. The motion must not require joints' effort that violate the actuator limits more than 5%, which is the margin allowed for the actuator at design time. We assume that there is a time-optimal motion planner, which can provide time-optimal trajectories with no torque violation. One formulation of such an optimization problem is:

$$\min_{\mathbf{x}_1,\mathbf{x}_2,\mathbf{u}} \quad T_f \quad \text{s.t.} \tag{1a}$$
$$\dot{\mathbf{x}}_1 = \mathbf{x}_2, \tag{1b}$$
$$\dot{\mathbf{x}}_2 = \mathbf{M}(\mathbf{x}_1)^{-1}(-\mathbf{C}(\mathbf{x}_1,\mathbf{x}_2) - \mathbf{G}(\mathbf{x}_1) - \boldsymbol{\tau}_{fric} + \mathbf{u}), \tag{1c}$$
$$\mathbf{q}_{min} \le \mathbf{x}_1 \le \mathbf{q}_{max}, \tag{1d}$$
$$-\dot{\mathbf{q}}_{min} \le \mathbf{x}_2 \le \dot{\mathbf{q}}_{max}, \tag{1e}$$
$$-\boldsymbol{\tau}_{min} \le \mathbf{u} \le \boldsymbol{\tau}_{max}, \tag{1f}$$

where T_f, $\mathbf{q}$, and $\boldsymbol{\tau}$ are total motion time, joint position, and joint torque respectively. $\mathbf{x}_1$, $\mathbf{x}_2$, are joint position and velocity as the state variable, and $\mathbf{u}$ is the joint torque as the input variable of the optimal control problem. In addition to joint position and velocity limits, the motion must respect the limits on torque which is related to the joint position and its derivatives through inverse dynamics defined by the mass matrix $\mathbf{M}$, the Coriolis and centrifugal vector $\mathbf{C}$, the gravity vector $\mathbf{C}$, and the friction torque $\boldsymbol{\tau}_{fric}$.

Given that state-of-the-art optimization-based approaches for such a problem cannot satisfy our computation-time requirement, we are interested in a learning-based solution that can closely predict the optimal motion after training in an offline phase, where the optimal motion planner is used as the ground truth to generate data (n motion planning queries) for training and validation.

Suppose we have a set of joint angles $\mathbf{q}$, denoted as $\mathcal{U}$, obtained by solving the constrained inverse kinematics problem of their corresponding poses in the task space of interest. The active learning algorithm samples the most informative set of joint angles $\mathcal{A}$ from $\mathcal{U}$. The set of a possible discrete payload is denoted as $\mathcal{M}$. We obtain $\mathbf{x}_i$ for $i = 1, \ldots, n$, by finding all possible combinations of two joint angles and a payload, which is $\mathcal{A}\times\mathcal{A}\times\mathcal{M}$. Then each query $\mathbf{x}_i = (\mathbf{q}_i, \mathbf{q}_f, \mu)$, where $\mu \in \mathcal{M}$. The optimal planner can then plan n trajectories and they are encoded through $\mathcal{T}$ to get $\boldsymbol{\omega}_i \in \mathbb{R}^{w\cdot d}$ for $i = 1, \ldots, n$, where w is the encoded dimension of each joint. The $\mathbf{x}_i$, $\boldsymbol{\omega}_i$ and the duration T_f^i of i-th trajectory, for $i = 1, \ldots n$, can form the data set $\mathcal{D} := \{\mathbf{X}, \mathbf{Y}\}$, where i-th row of $\mathbf{X}$ is $\mathbf{x}_i$ and i-th row of $\mathbf{Y}$ is $(\boldsymbol{\omega}_i, T_f^i)$. The desired predictor $\mathcal{P}$ is capable of dealing with the following learning task. After the training phase of $\mathcal{P}$ on dataset $\mathcal{D}$, given a query $\mathbf{x}_* \in \mathbb{R}^{2\cdot d+1}$ that satisfies the workspace and payload requirements, the predictor $\mathcal{P}$ is capable of predicting $\mathbf{y}_* \in \mathbb{R}^{w\cdot d+1}$, which is comprised of the encoding parameters $\boldsymbol{\omega}_*$ of the corresponding time-optimal trajectory and its duration T_f^*. The decoder $\mathcal{T}^{-1}$ exists and can convert $\boldsymbol{\omega}_*$ accompany with T_f^* back to an entire trajectory with acceptable deviations from the original trajectory. In the meanwhile, the capacity of $\mathcal{P}$ should be large enough to cope with the relatively large dataset brought by the enumeration of $\mathbf{q}_i$, $\mathbf{q}_f$ and $\mu \in \mathcal{M}$.

To make the learning problem well defined, we made the following three assumptions:

Assumption 1: The optimal trajectory in the vicinity of $\mathbf{x}_*$ exhibits continuity, indicating that the optimal solution does not undergo abrupt changes in the vicinity of the query point.

Assumption 2: The mapping $\mathcal{T}$ of the trajectory encoding is bijective and continuous. This, combined with assumption 1, implies that predicting the optimal trajectory is equivalent to predicting its encoded weights. Furthermore, the correlation between encoding weights can be measured by the distance between query points.

Assumption 3: The chosen constrained inverse kinematic scheme has a guaranteed existence and uniqueness of solution within the desired Cartesian sampling space. It is an essential requirement of performing task-oriented data sampling in Cartesian workspace that can be used for meaningful prediction.

3.1 Sparse Gaussian Process Regression

Gaussian Processes (GPs) assume there is a series of random variables, any finite number of which have a joint Gaussian distribution Williams and Rasmussen (2006). A GP can be completely described by a mean function $m(\mathbf{x})$ and a covariance function $k(\mathbf{x}, \mathbf{x}')$. Gaussian Process Regression (GPR) is a nonparametric method that uses the GP prior for Bayesian inference to obtain a predictive distribution corresponding to the query. With a zero prior of the mean function and Gaussian likelihood, the predictive distribution of the underlying scalar function f at a single point $\mathbf{x}_*$ is a Gaussian distribution $\mathcal{P}(\mathbf{x}_*|\mathcal{D}) \sim \mathcal{N}(\bar{f}_*, V(f_*))$, where

$$\bar{f}_* = \mathbf{k}_n^T(\mathbf{K}_{nn} + \sigma_n^2\mathbf{I})^{-1}\mathbf{t}, \tag{2a}$$
$$V(f_*) = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_n^T(\mathbf{K}_{nn} + \sigma_n^2\mathbf{I})^{-1}\mathbf{k}_n. \tag{2b}$$

We can regard $\mathbf{t} \in \mathbb{R}^n$ as a noisy realization of f with Gaussian noise, whose variance is σ_n^2. We can build a multi-output predictor $\mathcal{P}$ by fitting a scalar GPR to each output dimension. We use $\mathbf{K}_{nn}$ to denote this is a $n \times n$ matrix, whose i-th row j-th column is $k(\mathbf{x}_i, \mathbf{x}_j)$. $\mathbf{k}_n$ is used to denote a vector with n entries, whose i-th entry is $k(\mathbf{x}_*, \mathbf{x}_i)$. We adopted *Radial Basis Functions* (RBFs) as the covariance functions according to assumption 2. The expression of an RBF is

$$k(\mathbf{x}, \mathbf{x}') = exp\big\{-(\mathbf{x} - \mathbf{x}')^T\mathbf{W}(\mathbf{x} - \mathbf{x}')\big\}, \tag{3}$$

where $\mathbf{W}$ is a diagonal matrix. Its i-th diagonal entry is $1\,/\,l_i^2$, where l_i for $i = 1, \ldots, n$ are commonly called *kernel length* and are RBF hyperparameters. The hyperparameters in the covariance function $k(\mathbf{x}, \mathbf{x}')$ and noise σ_n can be determined by maximizing the log marginal likelihood $log\, p(\mathbf{t}) = log\, \mathcal{N}(\mathbf{t}|\mathbf{0}, \mathbf{K}_{nn} + \sigma_n^2\mathbf{I})$ of the GPR.

Because the payload among the input feature generates obvious sparsity that can be utilized to improve the speed and scalability of the model, we introduce sparse GPR in this work. The main challenge of sparse GPR lies in searching/learning for the inducing points denoted as $\mathbf{X}_m$ to build a low-rank approximation to the covariance matrix Liu et al. (2020). In this work, we regard the inducing points as a subset of training data $\mathbf{X}_m \in \mathbb{R}^{m\times(d+1)}$. The common methods for searching inducing points from $\mathcal{D}$

through active learning or variational inference can be found in Seeger et al. (2003) and Titsias (2009). Once $\mathbf{X}_m$ has been determined, the mean and variance functions of predictive distribution are

$$\bar{f}_* = \mathbf{k}_m^T(\sigma_n^2\mathbf{K}_{mm} + \mathbf{K}_{mn}\mathbf{K}_{nm})^{-1}\mathbf{K}_{mn}\mathbf{t}, \tag{4a}$$

$$V(f_*) = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_m^T\mathbf{K}_{mm}^{-1}\mathbf{k}_m + \sigma_n^2\mathbf{k}_m^T(\sigma_n^2\mathbf{K}_{mm} + \mathbf{K}_{mn}\mathbf{K}_{nm})^{-1}\mathbf{k}_m. \tag{4b}$$

Following a similar notation, we can regard $\mathbf{K}_{nm}$ as the matrix generated by the covariance function between the full training set and the inducing points. Furthermore, $\mathbf{k}_m$ is the vector generated by the covariance function between the test point and the inducing points. For the rest of the notation according to this rule, the meaning is straightforward. In equation (4), it is easily observed that the largest dimension of the matrix that needs to be inverted can be reduced from n to m compared to the full GPR. This property can reduce the computational complexity of training from $O(n^3)$ to $O(m^2n)$, variance prediction from $O(n^2)$ to $O(m^2)$ and mean prediction from $O(n)$ to $O(m)$.

3.2 FlexDMPs and DMP³

As clarified in the introduction section, we are interested in DMPs that retain acceleration information accurately. FlexDMPs is a trajectory encoding framework that retains the structural benefits of standard DMPs but can generate the trajectories differentiable up to the fourth degree, specially suited for robots with flexible joints Wahrburg et al. (2021). Following the FlexDMP approach, we can derive DMP^3, which (in contrast to standard DMPs) generate trajectories that are finite-jerk, i.e. continuous in acceleration. A DMP^3 is a third-order linear dynamical system, namely

$$\tau\dot{y_1} = y_2, \tag{5a}$$

$$\tau\dot{y_2} = y_3, \tag{5b}$$

$$\tau\dot{y_3} = \alpha_1(\alpha_2(\alpha_3(y_1^{goal} - y_1) - y_2) - y_3) + f(x), \tag{5c}$$

$$\tau\dot{x} = -\alpha_x x, \tag{5d}$$

$$f(x) = \frac{\sum_i^M \Psi_i(x)\omega_i}{\sum_i^M \Psi_i(x)} x(y_1^{goal} - y_1^{init}), \tag{5e}$$

$$\Psi(x) = e^{-\frac{1}{2\sigma_i^2}(x-c_i)^2}, \ i = (1 \dots M). \tag{5f}$$

The quantities y_1, y_2 and y_3 in equation (5) can be regarded as position, velocity and acceleration in our application. With y_1^{init} and y_1^{goal} we denote the initial and goal joint positions. Equation (5d) is conventionally regarded as the *canonical system*. The additional phase variable x is used to describe the extent to which the trajectory execution has been completed. Usually, $x(t = 0) = 1$ is used to initialize the phase variable and it will decay exponentially to 0, during the simulation of the autonomous system. The nonlinear forcing term $f(x)$ in equation (5c) is intended to encode the shape of the trajectory and has the expression in equation (5e), which is a weighted sum of M basis functions $\Psi_i(x)$. The term $x(y_1^{goal} - y_1^{init})$ is designed to handle the spatial scaling when the learned forcing function is used to generate a trajectory with a modified goal and/or initial positions. It also acts as the diminishing term, when x is 0 or $y_1^{goal} = y_1^{init}$. Univariate RBFs are commonly used basis function as shown in equation (5f). The determination of the parameters c_i and σ_i for each RBF will depend on the evolution of the phase variable, and therefore there is no unique method. Overall, RBFs need to be scattered evenly over the entire trajectory depending on the time scale. Here, we use $c_i = e^{-\alpha_x\Delta_t}$, where Δ_t are discrete-time points linearly spaced according to the number of RBFs, and $\sigma_i^2 = \frac{\alpha_x c_i}{2hM^2}$, where we regard h as a width factor that needs to be tuned. When trajectories have strong non-linearities, it is usually beneficial to use a smaller σ_i, which naturally corresponds to a larger number of RBFs by this heuristic method. The parameter τ acts as a time-scaling factor. It will accelerate the execution of the motion when $\tau \in (0, 1)$ and slow down when $\tau > 1$.

With appropriate gain coefficients α_1, α_2 and α_3, the target state becomes a stable attractor of this third-order autonomous system (5a)-(5c), so there is guaranteed convergence to the target state. FlexDMPs suggest adopting the reasoning used for standard DMPs and making the autonomous system critically damped to ensure monotonic convergence to the goal y_1^{goal} for $f(x) = 0$ Ijspeert et al. (2013). For DMP^3 this leads to the following procedure of determining the coefficients α_i for $i = 1, 2, 3$. The characteristic polynomial of the system described by (5a)-(5c) is

$$P(s) = s^3 + \frac{\alpha_1}{\tau}s^2 + \frac{\alpha_1\alpha_2}{\tau^2}s + \frac{\alpha_1\alpha_2\alpha_3}{\tau^3}. \tag{6}$$

By comparing equation (6) and the characteristic polynomial of a third-order system with a three-fold pole at $-p/\tau$, which is

$$P(s) = (s + \frac{p}{\tau})^3, \\ = s^3 + 3\frac{p}{\tau}s^2 + 3\frac{p^2}{\tau^2} + \frac{p^3}{\tau^3}, \tag{7}$$

we infer the gain coefficients as

$$\alpha_1 = 3p, \ \alpha_2 = p, \ \alpha_3 = \frac{1}{3}p. \tag{8}$$

Therein, p is a parameter that determines the distance of the poles lying on the left real axis from the origin.

Because of the linear dependency of the weights, ω_i, and the basis function $\Psi_i(x)$, the weights of a demonstrated trajectory are commonly determined by *locally weighted regression* Ijspeert et al. (2013). The target forcing term f_{target}^k at time step k can be derived from equation (5c) by eliminating the states y_2 and y_3 by substituting equations (5a) and (5b). This results in

$$f_{target}^k = \alpha_1\alpha_2\alpha_3(y_{demo}^k - y_1^{goal}) + \tau\alpha_1\alpha_2\dot{y}_{demo}^k + \tau^2\alpha_1\ddot{y}_{demo}^k + \tau^3\dddot{y}_{demo}^k, \tag{9}$$

where y_{demo} is the trajectory of joint positions from a demonstration. For a multidimensional trajectory encoder $\mathcal{T}$, we can fit a DMP^3 for each dimension and let them share the same canonical system. To decode the weights into trajectories, we only need to simulate this third-order autonomous system starting from the initial states.

3.3 Active Learning with Mutual Information Criterion

In a pick-and-place application, the sampling spaces can be boxes encompassing the in- and out-feed regions as

shown in Fig. 1b. To select s points in the joint space with the most significant reduction of uncertainty in the constrained Cartesian space, a good criterion is to maximize the *mutual information* between the chosen point set $\mathcal{A}$ from the universal set $\mathcal{U}$ and the rest of the candidates $\mathcal{U} \setminus \mathcal{A}$ Krause et al. (2008). The mutual information is defined as $I[x, y] = H[x] - H[x|y]$, where $H[\cdot]$ denotes the entropy of a random variable. Thus, the maximum of mutual information $I(\mathbf{Q}_\mathcal{A}, \mathbf{Q}_{\mathcal{U}\setminus\mathcal{A}})$ is equivalent to find the solution of

$$\mathcal{A}^* = \underset{\mathcal{A}\subseteq\mathcal{U}:|\mathcal{A}|=s}{\operatorname{argmax}} \; H(\mathbf{Q}_\mathcal{A}) - H(\mathbf{Q}_{\mathcal{U}\setminus\mathcal{A}}|\mathbf{Q}_\mathcal{A}). \tag{10}$$

We use $\mathbf{Q}_\mathcal{A}$ and $\mathbf{Q}_{\mathcal{U}\setminus\mathcal{A}}$ to refer to sets of random variables of joint angles at the locations $\mathcal{A}$ and $\mathcal{U} \setminus \mathcal{A}$. This criterion will lead to an intuitively central placement of query points that unlike the entropy-based method will have "wasted information" by putting a lot of query points on the boundary of $\mathcal{U}$. However, the problem (10) has an intractable complexity (is NP-complete), which makes it difficult to use directly in real-world applications (Krause et al. (2008) Theorem 2). Thus, we resort to a one-step greedy approximation to find a point that reduces the objective function the most according to

$$a^* = \underset{a}{\operatorname{argmax}}\, I(\mathbf{Q}_{\mathcal{A}\cup a}, \mathbf{Q}_{\mathcal{U}\setminus(\mathcal{A}\cup a)}) - I(\mathbf{Q}_\mathcal{A}, \mathbf{Q}_{\mathcal{U}\setminus\mathcal{A}}), \tag{11a}$$

$$= \underset{a}{\operatorname{argmax}}\, H(\mathbf{q}_a|\mathbf{Q}_\mathcal{A}) - H(\mathbf{q}_a|\mathbf{Q}_{\bar{\mathcal{A}}}), \tag{11b}$$

where $\bar{\mathcal{A}}$ means $\mathcal{U} \setminus (\mathcal{A} \cup a)$. We can interpret the first term in (11b) as finding the next joint angle $\mathbf{q}_a$ with the maximum uncertainty given the current $\mathbf{Q}_\mathcal{A}$. The second term is to find the next $\mathbf{q}_a$ with the lowest uncertainty given $\bar{\mathcal{A}}$, which can be regarded as a centering behavior among the remaining candidates. The tradeoff between these two terms will produce a series of points that are nearly evenly distributed in $\mathcal{U}$ without pushing a large number of points to the boundary of $\mathcal{U}$ as in the entropy-based approach.

To iteratively update the set $\mathcal{A}$, the objective function (11b) can be evaluated for each candidate in $\mathcal{U}\setminus\mathcal{A}$, then the point with maximum value should be added into $\mathcal{A}$. The expression of the objective function can be derived based on the differential entropy of the predictive distribution of GPR, resulting in $H(\mathbf{x}_*|\mathcal{D}) = \frac{1}{2}log(2\pi eV^2(f_*))$, where the term $V(f_*)$ has been shown in equation (2b). Equation (11b) can be rewritten explicitly as

$$a^* = \underset{a}{\operatorname{argmax}} \frac{k(\mathbf{q}_a, \mathbf{q}_a) - \mathbf{k}_\mathcal{A}^T(\mathbf{K}_\mathcal{A} + \sigma_n^2\mathbf{I})^{-1}\mathbf{k}_\mathcal{A}}{k(\mathbf{q}_a, \mathbf{q}_a) - \mathbf{k}_{\bar{\mathcal{A}}}^T(\mathbf{K}_{\bar{\mathcal{A}}} + \sigma_n^2\mathbf{I})^{-1}\mathbf{k}_{\bar{\mathcal{A}}}}. \tag{12}$$

According to the notation of sparse GPR, $\mathbf{K}_\mathcal{A}$ is the matrix generated by the covariance function with $\mathbf{Q}_\mathcal{A}$ and itself as input; $\mathbf{k}_\mathcal{A}$ is the vector generated by the covariance function with $\mathbf{q}_a$ and $\mathbf{Q}_\mathcal{A}$. Similar reasoning applies to $\mathbf{K}_{\bar{\mathcal{A}}}$ and $\mathbf{k}_{\bar{\mathcal{A}}}$. In Krause et al. (2008), it is shown that this greedy approximation will generate the solution with at least 63% suboptimality of the problem (10). This allows us to obtain a set of joint angles, as described in the problem statement.

3.4 Torque Violation Avoidance

A time-optimal trajectory exploits the defined limits of the system, and therefore the smallest deviation from the optimal trajectory can lead to exceeding the limits. To reduce torque limit violations, we performed the training with slightly reduced torque limits. Such a "backoff" trick can be effective although it is inconvenient in practice, as determining the magnitude of the reduction is ultimately problem dependent and may require some trial and error. Such parameter tuning is however also needed in kinematic planners when setting acceleration and jerk limits. Nonetheless, there might still be outliers that violate the limits, and should therefore be post-processed. We used the simple post-processing approach of iteratively prolonging the trajectory until the violation is within the acceptable range. The prolonging can be achieved through adjusting the time-scaling parameters τ in equation (5).

4. RESULTS

4.1 Box to Box with fixed Orientation

A first experiment comprised two box regions in the Cartesian space as shown in Fig. 1b, sampled with fixed end-effector orientation. Despite their simplicity, such box regions can in practice cover real-world applications of industrial robots such as item-picking. The selected end-effector orientation resulted in a protracting tool being in parallel with the Z axis of the world coordinate system which was perpendicular to the table surface. A unique solution of inverse kinematics, as needed by assumption 3, is guaranteed through the choice of a suitable configuration parameter. The selected box here is a cube with 22 cm length. There are 64 evenly distributed pose samples per box, depicted with the coordinate frames in Fig. 1b, amounting to a full enumeration of 4096 pairs of $\mathbf{q}_i$ and $\mathbf{q}_f$. The manipulator used both in offline and lab experiments is an ABB IRB1100-4-058 with a maximum payload of 4 Kg. The payload mass range of 0 to 4 Kg was discretized with a step of 0.5 Kg, resulting in a total number of 36864 queries n in the dataset $\mathcal{D}$. The gain coefficient p in DMP^3 was set to be 11. The numerical integration of DMP^3 was realized by the Runge-Kutta fourth-order method, as we found that the precision of the integration plays a big role in the peak torque violation. We used 30 basis functions so that $\mathcal{T}$ retains sufficient dynamic features of time-optimal trajectory. Thus the learning problem becomes $\mathcal{P} : \mathbb{R}^{13} \rightarrow \mathbb{R}^{181}$. We found to get a relatively precise prediction that can satisfy the requirements at the beginning of section 3, we have to set the number of inducing points m sufficiently large. Thus we sampled 8500 data points randomly from $\mathcal{D}$ as m inducing points and let output channels share the same group of $\mathbf{X}_m$. Then the hyperparameters were searched through maximizing the marginal likelihood. We implemented and evaluated the methods in Seeger et al. (2003) and Titsias (2009). However, we found that they did not outperform random sampling, which we believe is mainly due to the following two reasons. First, a large number of inducing points would nullify the advantages gained from selecting more informative data. Second, there is no dominant hidden structure that can be exploited by the sparse approximation in the learning problem of a single payload. However, this approach results in highly optimal motions without exceeding the torque beyond the allowed limits, as shown in Table 1. The test set comprises 2025 trajectories with randomly sampled queries, namely $\mathbf{q}_i, \mathbf{q}_f$ anywhere in each box (i.e., not the samples used in the training), and a

Fixed Ori.	Time sub-opt (ms)	Sub-opt (%)	Torque Vio. (%)
No Intervention	1.7 (4.0)	0.31 (1.0)	3.5 (7.1)
+ 3 % Backoff	5.8 (15)	2.0 (4.7)	0.69 (4.4)
+ Prolong	-	-	-
Variable Ori.	Time sub-opt (ms)	Subopt. (%)	Torque Vio. (%)
No Intervention	5.033 (42.95)	1.322 (15.35)	3.473 (11.22)
+ 7 % Backoff	14.78 (61.12)	4.150 (21.95)	0.010 (10.50)
+ Prolong	14.79 (61.12)	4.155 (21.95)	0.009 (4.134)

Table 1. Training results of fixed orientation and variable orientation scenarios. Data outside brackets are mean values, those inside brackets are the maximum values. The optimal motion duration was used as the ground-truth to calculate the motion time sub-optimality.

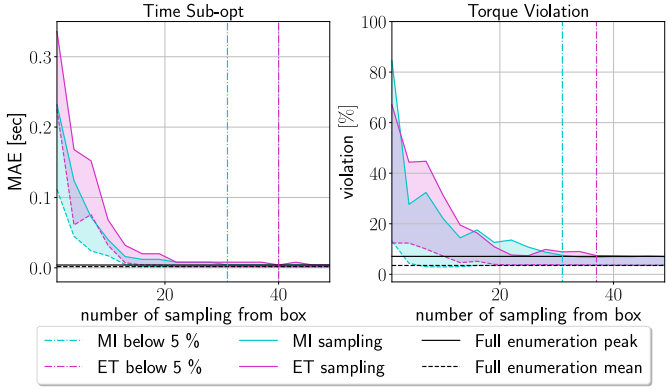


Fig. 2. Approximation test by active learning. MI: mutual information, ET: entropy. The dashed-dotted lines indicate that the approximation performance drop is below 5 % compared to the full enumeration. For each x-axis value, the evaluation was performed on 2025 randomly selected trajectories. The solid lines indicate the peak values and the dashed lines indicate the mean values. In pool-based sampling with fixed hyperparameters, there is no stochastic behavior, so the predictor was trained only once for each x-value.

random payload from the discretized set. The motion time sub-optimality of predicted trajectories was evaluated by mean absolute error (MAE).

Although the average violation was only 3.5%, a 3% backoff was applied to bring the maximum violations of the torque limit below the desired 5%. Since this measure was enough in bringing the violation into the acceptable range, no trajectory prolongation was needed.

Since it is still possible to fully enumerate the compositions of $\mathbf{q}_i$, $\mathbf{q}_f$ in this scenario, we performed a comparison between entropy-based and mutual information-based active learning and compared the approximated performance with the full enumeration. The sampling pool $\mathcal{U}$ is a set of 64 joint angles for each box. In this work, sampling through active learning was performed with fixed kernel hyperparameters. After finishing the sampling to get a $\mathcal{A}$ for each box, we followed the same steps as in the previous description to complete the data generation, training and evaluation. The results in Fig. 2 show that the mutual information criterion in the box-to-box setting is superior to the entropy-based approach. The mutual information approach achieved 5% performance deterioration with respect to the full enumeration at sample number 31 in both graphs, while the entropy approach achieves this criterion by 40. At the same time, we can observe that the mean value of the mutual information approach decreases much faster than the entropy-based approach. This advantage will prove valuable in the more complex variable orientation scenario.

4.2 Box to box queries with variable orientation

In a second experiment, we varied the orientation of the end-effector to address the requirement of industrial applications involving angled pick-and-place. Each pose from the previous experiment was augmented by adding rotations of $\{10°, -10°\}$ around Y and X axes, producing a sampling pool with $|\mathcal{U}| = 1225$ joint angles per box. This not only increased the number of samples, but also considerably increased the span of the sampled joint space compared to the fixed orientation case. The number of chosen joint angles per box $|\mathcal{A}|$ is 70, which was determined by inspecting the training progress of active learning. Following the same steps and settings as for the fixed orientation case, the results shown in the Table 1 were obtained. It should be noted that the worst case 10.5% of torque violation after 7% backoff is due to one outlier among 2025 trajectories. This outlier was prolonged by 8% to bring it in the acceptable 5% torque limit violation. An example of the predicted trajectories for a 4 Kg payload is shown in Fig. 3 along with the trajectory generated from the optimization-based planner. Since joints have different torque limits, to enhance the visualization, the torque utilization is plotted instead of torque, that is simply the normalized torque value according to the corresponding limit of each joint. The torque margin of the predicted trajectory brought about by the backoff trick can be seen in the utilization plot. It is worth noting that at 0.2 seconds joint 3 crosses zero velocity and the torques exhibit a highly non-linear behavior due to the joint friction. The predicted trajectory closely follows the optimal trajectory, reflecting the ability of our planner to accurately account for dynamics through learning.

The proposed solution can be extended to different workspace regions, tasks, or optimization criteria such as energy minimization. Since the ground truth in this work is the optimal trajectory, experimentation on the real robot does not add new information for evaluating the method further. Nonetheless, we performed a lab test with an ABB IRB1100-4-058, showed in the accompanying video [2], to visually showcase the type of time-optimal trajectories studied in this work, and to demonstrate the considerable reduction of the computation times.

4.3 Comparison with Ruckig

To further demonstrate the performance of the proposed planner, we made a comparison to an optimal kinematic planner. Ruckig is a recent and popular motion planning library, that can generate jerk-limited trajectories that are time-optimal with respect to kinematic constraints Berscheid and Kröger (2021). As mentioned in the introduction, acceleration and jerk have no explicit electro-mechanical limitations, and one has to set them somehow

[2] https://youtu.be/GGJU7nfYOBY

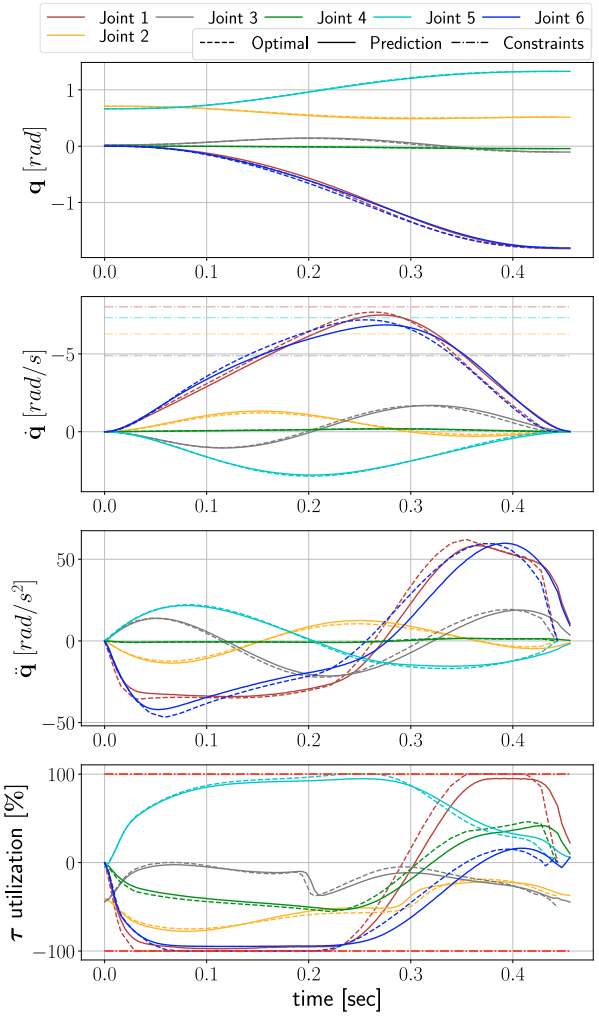


Fig. 3. Comparison of optimal and predicted trajectories. The constraints for $\mathbf{q}$ were omitted, because none of them is tight. The utilisation is the ratio of the current value relative to the limits.

through trial and error, to get an acceptable performance while not violating actuator limits. We attempted to set the acceleration limit of each joint by finding a motion resulting in maximum torques. For instance, for joint 1 of the robot arm used in this work, a 180 degree joint 1 rotation in a fully stretched horizontal configuration with maximum payload was chosen. The acceleration limit was then iteratively reduced from a high value until the maximum torque violation was below the allowed 5% limit. This worst-case approach, however, severely reduced the time-optimality of the trajectories. Therefore we increased the acceleration limits by trial and error to enhance the motion speed until average torque violation for our set of evaluation queries remained in the required range. The results of both Ruckig and our planner on the fixed orientation box to box test set, which has 225 trajectories for each payload, are shown in Fig. 4. As depicted by the motion duration plots, despite the manual tuning of acceleration limits for better performance, the time-optimality is drastically lower than the trained planner. As expected, the optimal and learned planners produce slower motions for higher payloads, as they consider the dynamics of the motion and actuator limits. The motion duration box plot on the right shows the time sub-optimality of the two methods relative to the optimal motion for all payloads. From the torque violation plots, it can be noted that although both planners have an average violation that is below our required 5%, there are several outliers where the Ruckig trajectories have considerably violated above 5%, highlighting the challenge of using kinematic planners for high-performance motion.

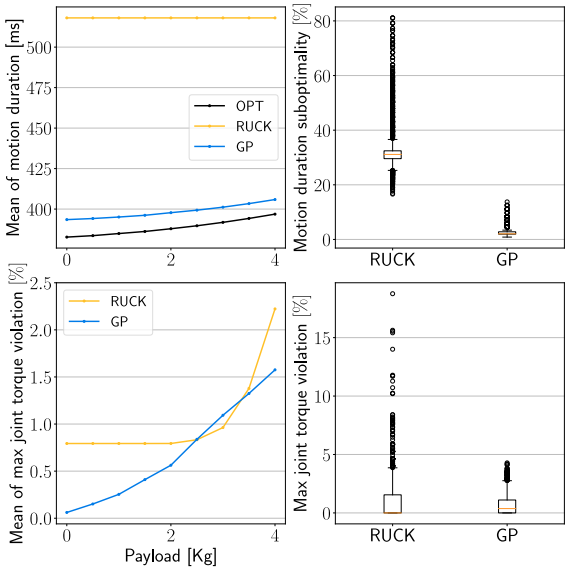


Fig. 4. Comparison of movement duration and torque violation. OPT, RUCK and GP stand for the optimal planner, the Ruckig planner and our learning-based planner.

Fig. 5 summarizes the achieved results with respect to our requirement for a time-optimal motion planner that plans in a few tens of milliseconds. In terms of computation time, as expected, optimization-based planning can take several hundreds of milliseconds to some seconds. Kinematic planners such as Ruckig can produce trajectories in sub-millisecond range. However, as already established, the time-optimality of kinematic planners, when restricted to the actual limits of the actuators, can only be drastically sub-optimal. Our proposed planner has produced close to optimal trajectories in about 20 milliseconds on average. Note that the planner was implemented in Python, and the computation time can considerably be shortened by implementing it in a more performant language.

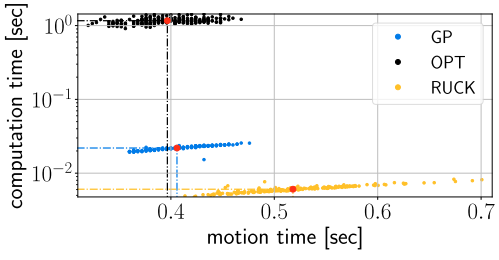


Fig. 5. Motion times and compute times for the three planners in the case of 4 Kg payload. The red dots show the mean value.

5. CONCLUSION

This work proposed a combination of DMP^3, sparse GPR, and active learning, to efficiently train a motion planner in a sub-space of a 6-DoF manipulator's workspace. The trained planner produces, in a few tens of milliseconds, highly time-optimal motions for variable robot payloads that outperform motions generated by optimal kinematic planners while respecting actuator torque limits. Of course, our proposed method requires an offline training phase for a workspace/application, whereas the optimal planner and kinematic planners are general-purpose and runtime-ready. Nonetheless, for many industrial applications with repetitive motions running for days, it can be acceptable to have an offline training phase, to achieve near-optimal motion time in a few 10s of milliseconds and therefore considerably enhance productivity. The near-optimal trajectory can also be used to warm-start optimization-based planners and reduce their computation time. Future work can include larger/different workspace sub-spaces and other motion requirements such as as energy-minimization or application-specific constraints. Generalization to queries outside of the sampled sub-space is another topic to studied.

REFERENCES

Berscheid, L. and Kröger, T. (2021). Jerk-limited real-time trajectory generation with arbitrary target states. *arXiv preprint arXiv:2105.04830*.

Calinon, S. (2016). A tutorial on task-parameterized movement learning and retrieval. *Intelligent service robotics*, 9(1), 1–29.

Conkey, A. and Hermans, T. (2019). Active learning of probabilistic movement primitives. In *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*, 1–8. IEEE.

Deniša, M., Gams, A., Ude, A., and Petrič, T. (2015). Learning compliant movement primitives through demonstration and statistical generalization. *IEEE/ASME transactions on mechatronics*, 21(5), 2581–2594.

Fanger, Y., Umlauft, J., and Hirche, S. (2016). Gaussian processes for dynamic movement primitives with application in knowledge-based cooperation. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 3913–3919. IEEE.

Forte, D., Gams, A., Morimoto, J., and Ude, A. (2012). On-line motion synthesis and adaptation using a trajectory database. *Robotics and Autonomous Systems*, 60(10), 1327–1339.

Girgin, H., Pignat, E., Jaquier, N., and Calinon, S. (2020). Active improvement of control policies with bayesian gaussian mixture model. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 5395–5401. IEEE.

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

Krause, A., Singh, A., and Guestrin, C. (2008). Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, 9(2).

Kulak, T., Girgin, H., Odobez, J.M., and Calinon, S. (2021). Active learning of bayesian probabilistic movement primitives. *IEEE Robotics and Automation Letters*, 6(2), 2163–2170.

Laumond, J.P. et al. (1998). *Robot motion planning and control*, volume 229. Springer.

Lembono, T.S., Paolillo, A., Pignat, E., and Calinon, S. (2020). Memory of motion for warm-starting trajectory optimization. *IEEE Robotics and Automation Letters*, 5(2), 2594–2601.

Liu, H., Ong, Y.S., Shen, X., and Cai, J. (2020). When gaussian process meets big data: A review of scalable gps. *IEEE transactions on neural networks and learning systems*, 31(11), 4405–4423.

Maeda, G., Ewerton, M., Osa, T., Busch, B., and Peters, J. (2017). Active incremental learning of robot movement primitives. In *Conference on Robot Learning*, 37–46. PMLR.

Pervez, A. and Lee, D. (2018). Learning task-parameterized dynamic movement primitives using mixture of gmms. *Intelligent Service Robotics*, 11(1), 61–78.

Qureshi, A.H., Simeonov, A., Bency, M.J., and Yip, M.C. (2019). Motion planning networks. In *2019 International Conference on Robotics and Automation (ICRA)*, 2118–2124. IEEE.

Saveriano, M., Abu-Dakka, F.J., Kramberger, A., and Peternel, L. (2021). Dynamic movement primitives in robotics: A tutorial survey. *arXiv preprint arXiv:2102.03861*.

Saveriano, M. and Lee, D. (2018). Incremental skill learning of stable dynamical systems. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 6574–6581. IEEE.

Seeger, M.W., Williams, C.K., and Lawrence, N.D. (2003). Fast forward selection to speed up sparse gaussian process regression. In *International Workshop on Artificial Intelligence and Statistics*, 254–261. PMLR.

Siciliano, B., Sciavicco, L., Villani, L., and Oriolo, G. (2010). Robotics: modelling, planning and control. *New YorN, NY, USA: Springer*, 415–418.

Titsias, M. (2009). Variational learning of inducing variables in sparse gaussian processes. In *Artificial intelligence and statistics*, 567–574. PMLR.

Vijayakumar, S. and Schaal, S. (2000). Locally weighted projection regression: An o (n) algorithm for incremental real time learning in high dimensional space. In *Proceedings of the seventeenth international conference on machine learning (ICML 2000)*, volume 1, 288–293. Morgan Kaufmann.

Wahrburg, A., Guida, S., Enayati, N., Zanchettin, A.M., and Rocco, P. (2021). Flexdmp–extending dynamic movement primitives towards flexible joint robots. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 7592–7598. IEEE.

Weitschat, R., Haddadin, S., Huber, F., Albu-Scha, A., et al. (2013). Dynamic optimality in real-time: A learning framework for near-optimal robot motions. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 5636–5643. IEEE.

Williams, C.K. and Rasmussen, C.E. (2006). *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA.