IEEE Sensors Journal, 2015. DOI 10.1109/JSEN.2015.2455814

Sensor Calibration and Hysteresis Compensation with Heteroscedastic Gaussian Processes

Sebastian Urban*, Marvin Ludersdorfer†, Patrick van der Smagt*†
*Institut für Informatik VI, Technische Universität München, Boltzmannstr. 3, 85748 Garching, Germany
†fortiss—An-Institut der Technischen Universität München, Guerickestrasse 25, 80805 München, Germany

***Abstract*—We deal with the problem of estimating the true measured scalar quantity from the output signal of a sensor that is afflicted with hysteresis and noise. We use a probabilistic, non-parametric sensor model based on heteroscedastic Gaussian processes, which is trained using a dataset of sensor output and ground truth pairs. The inference problem is formulated as state estimation in a dynamical system.**

We exploit the low dimensionality of the latent state space of the sensor to perform *exact* probabilistic inference of the measured quantity from a time series of the sensor's output. Compared to the state-of-the-art assumed density filtering algorithm for Gaussian processes, which analytically approximates the posterior by a normal distribution during inference, our method reduces the prediction error by 33% on a dataset obtained from a novel flexible tactile sensor based on carbon-black filled elastomers.

The proposed model can be applied, but is not limited, to any sensor for which the Preisach model of hysteresis holds. The use of probabilistic modeling and inference not only provides a most likely estimate of the measured quantity but also the corresponding confidence interval.

I. Introduction

Consider a sensor that measures a time-varying, scalar quantity q_t and outputs a d-dimensional signal $\boldsymbol{x}_t$ that encodes q_t; for example an artificial skin that measures pressure. Typically one prefers a sensory signal free of nonlinearities and hysteresis [1], [2]; however, other requirements, such as physical flexibility, often collide with this property. The example we will be using throughout the article is the DLR artificial skin [3], [4], an elastomer-based tactile sensor shown in Figure 4. When an elastomer is compressed and released, its contraction and elongation curves in the stress-strain space do not coincide due to energy lost as heat [5]. Thus, due to the inherent internal friction of the elastomer [6], its response is afflicted with hysteresis.

In the past many approaches to quantify and compensate for hysteresis have been developed and applied. The Preisach model [7] is one of the most popular and widely used hysteresis models. It assumes that the response of a hysteretic system is determined by the weighted superposition of elementary *relay hysterons*, which respond independently of each other to the system's input. A relay hysteron is a binary operator that has two individual switching thresholds, one for the transition from the "off" to "on" state and another one for the inverse transition. If the input value enters the region between these thresholds, the relay hysteron maintains its previous response. To obtain a smooth response, a large number of hysterons with evenly distributed thresholds are employed. Since the state of individual hysterons is not directly observable, the Preisach model can be interpreted as a latent variable model with a high-dimensional, binary latent state space and a simple, deterministic transition function for each latent variable. Davino *et al.* [8] and Oppermann *et al.* [9] use an inverse to their respective Preisach models to compensate hysteresis in a magnetic field sensor and a force sensor, respectively. In the context of control Visone [10] applies an inverted Preisach model to sensors and actuators afflicted with hysteresis; thus a linear system is obtained for which well-established control schemes exist.

In this work we employ a different approach to model hysteresis. Like the Preisach model we use a latent variable model; however instead of employing a large set of binary latent variables with simple transition functions we use only *one* real-valued latent state and a powerful, non-parametric transition function. More formally, let $\boldsymbol{x}_t$ depend on the history of q; that is, $\boldsymbol{x}_t = \boldsymbol{f}_t(q_1, q_2, \ldots, q_t)$. For each time step t we introduce a latent variable h_t with a temporal evolution given by a transition function g such that $h_t = g(q_t, h_{t-1})$ and the sensor's output is governed solely by the hidden state. We thus have $\boldsymbol{x}_t = \boldsymbol{f}(h_t)$ for a suitable measurement function $\boldsymbol{f}$. Inference of the measured quantity from the sensor's output is equivalent to estimating the internal state of a nonlinear dynamical system from noisy observations. If a parametric model of the sensor is available the transition and measurement functions can be derived from it. Here, however, we assume that no model is available. Instead a dataset of the measured quantity and the sensor's corresponding output signal is at our disposal. Hence we use Gaussian process (GP) regression [11] to estimate both functions from the available data.

Previous work on state estimation assumes a moderate to high dimensional latent state dimensionality. The Extended Kalman Filter (EKF) [12] linearizes the transition and measurement functions by means of a Taylor series expansion and applies the Kalman filter [13] on the linearized problem. The Unscented Kalman Filter (UKF) [14] was developed to improve upon the EKF in the case where the functions were poorly represented by a linear approximation; it preserves the nonlinear function but uses deterministic sampling to propagate Gaussian distributions through the nonlinearity. In their GP-BayesFilters framework, Ko *et al.* [15], [16] combined the EKF and UKF with a GP regression model for the transition and measurement functions. The GP-ADF

Corresponding author: Sebastian Urban (surban@tum.de).

framework by Deisenroth *et al.* [17] analytically propagates a normal distribution through a GP and approximates the posterior using a Gaussian that matches the true posterior's mean and variance.

Since our work deals with sensors which measure a scalar quantity (or a set of scalar quantities that are assumed to be independent), the latent state space is of low dimensionality; this allows us to eliminate the approximation error by using exact inference methods. During training we estimate suitable values for the latent variables from the measured quantity and the sensor's corresponding output. Optionally the estimated values for the latent variables may be refined using the expectation maximization (EM) algorithm; however in our tests the initial estimates already yield excellent results.

II. Preliminaries

A. Gaussian Process Regression

A Gaussian process (GP) [11] is a non-parametric tool for learning scalar regression functions from sample data. A GP describes a stochastic process in which the random variables, in this case the outputs of the modeled function, are jointly Gaussian distributed. Consider the set of input and output data $\mathcal{D} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$. While the true value of each output point is a deterministic function of the input value, the corresponding observation is afflicted with Gaussian noise. This can be modeled as $y_i = f(\boldsymbol{x}_i) + \epsilon$, where $\epsilon \sim \mathcal{N}(0, r_i^2)$. Inputs, outputs and noise levels are aggregated into $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n]$, $\boldsymbol{y} = [y_1, \ldots, y_n]$, and $\boldsymbol{r} = [r_1, \ldots, r_n]$, respectively. The joint distribution over the outputs $\boldsymbol{y}$ is a zero-mean multivariate Gaussian,

$$p(\boldsymbol{y}) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{K}_X + \boldsymbol{R}),$$

where $\boldsymbol{K}_X$ is the kernel matrix with elements $K_{X_{ij}} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and $\boldsymbol{R} = \operatorname{diag}(\boldsymbol{r})$. The kernel function $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is a measure of similarity between two inputs. A popular choice is the squared exponential kernel given by

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sigma_f^2 \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2}{2L}\right),$$

where σ_f^2 is the signal variance and L is called the length scale of the kernel; it determines how fast the data varies if $\boldsymbol{x}$ changes. Other choices, such as the linear kernel $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i^T \boldsymbol{x}_j$, are also possible.

Prediction is done by conditioning the GP upon the observed training points. The resulting conditional distribution for a test input $\boldsymbol{x}^*$ with noise $\boldsymbol{r}^*$ is given by

$$p(y^* \mid \boldsymbol{x}^*, \mathcal{D}) = \mathcal{N}(\mathrm{GP}^\mu(\boldsymbol{x}^*, \mathcal{D}), \mathrm{GP}^\sigma(\boldsymbol{x}^*, \mathcal{D})),$$

with mean

$$\mathrm{GP}^\mu(\boldsymbol{x}^*, \mathcal{D}) = \boldsymbol{k}^{*T} \left[\boldsymbol{K}_X + \boldsymbol{R}\right]^{-1} \boldsymbol{y}$$

and variance

$$\mathrm{GP}^\sigma(\boldsymbol{x}^*, \mathcal{D}) = k(\boldsymbol{x}^*, \boldsymbol{x}^*) + \boldsymbol{R}^* - \boldsymbol{k}^{*T} [\boldsymbol{K}_X + \boldsymbol{R}]^{-1} \boldsymbol{k}^*,$$

where $\boldsymbol{k}^* = [k(\boldsymbol{x}^*, \boldsymbol{x}_1), \ldots, k(\boldsymbol{x}^*, \boldsymbol{x}_n)]$ and $\boldsymbol{R}^* = \operatorname{diag}(\boldsymbol{r}^*)$. The predictive variance depends on the variance of the training data and the dissimilarity, as measured by the kernel function, between the training data and the test input. Intuitively, a GP will assign a high predictive variance to a test input that is "far away" from the training samples because there is less evidence in its vicinity to rely upon.

A GP can either be homo- or heteroscedastic. In the homoscedastic case the noise level is assumed to be constant across all training and test points, thus $r_i = \sigma_n^2$. The parameters $\boldsymbol{\theta} = \{\sigma_n, \sigma_f, L\}$ are called hyper-parameters of the GP and are learned from the training data by maximizing the marginal likelihood of the training outputs given the inputs, $\theta_{\max} = \operatorname{argmax}_\theta \log p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta})$, using standard numerical optimization methods.

B. Heteroscedastic GP Regression

If the homoscedastic model is applied to data where the amount of noise varies between input points, i.e., heteroscedastic data, it will choose σ_n^2 according to the average noise level. As can be seen from Figure 2a, the GP will thus locally over- or underestimate the accuracy of its predictions. While the prediction uncertainty is of secondary importance for many applications, it becomes essential when the predictions of the GP are processed further using a probabilistic model.

A heteroscedastic GP requires a method to estimate the noise r_i for each training point and predict it for test inputs. Goldberg *et al.* [18] place an independent GP over the logarithms of the noise levels, denoted as $z(\boldsymbol{x}) = \log r(\boldsymbol{x})$. This z-process uses a different set of hyper-parameters $\boldsymbol{\theta}_z$ and a separate covariance function k_z. The noise levels $\boldsymbol{r}$ and $\boldsymbol{r}^*$ are now described by a probabilistic model and thus the predictions of the heteroscedastic GP can be calculated by marginalization, thus

$$\begin{aligned} &p(y^* \mid \boldsymbol{x}^*, \mathcal{D}) \\ &\quad = \iint p(y^* \mid \boldsymbol{x}^*, \boldsymbol{z}, z^*, \mathcal{D})\, p(\boldsymbol{z}, z^* \mid \boldsymbol{x}^*, \boldsymbol{X})\, \mathrm{d}\boldsymbol{z}\, \mathrm{d}z^*. \end{aligned} \quad (1)$$

Since analytic calculation of this integral is intractable, Goldberg *et al.* proposed to obtain Monte Carlo samples $\{(\boldsymbol{z}_1, z_1^*), \ldots, (\boldsymbol{z}_k, z_k^*)\}$ from $p(\boldsymbol{z}, z^* \mid \boldsymbol{x}^*, \boldsymbol{X})$ and approximate the integral by $\frac{1}{k} \sum_{j=1}^k p(y^* \mid \boldsymbol{x}^*, \boldsymbol{z}_j, z_j^*, \mathcal{D})$.

To avoid the time-intensive sampling procedure, Kersting *et al.* [19] proposed to further approximate $p(y^* \mid \boldsymbol{x}^*, \mathcal{D})$ by replacing the marginalization (1) with the most likely point estimate $(\tilde{\boldsymbol{z}}, \tilde{z}^*)$. The predictive distribution is thus given by $p(y^* \mid \boldsymbol{x}^*, \mathcal{D}) = p(y^* \mid \boldsymbol{x}^*, \tilde{\boldsymbol{z}}, \tilde{z}^*, \mathcal{D})$. This approximation is good when most of the probability mass of $p(\boldsymbol{z}, z^* \mid \boldsymbol{x}^*, \boldsymbol{X})$ is located around $(\tilde{\boldsymbol{z}}, \tilde{z}^*)$. Since both GPs are interdependent, training must be performed using an EM-like procedure which iteratively trains the data GP and z-process until convergence is achieved.

C. Bayesian Networks

A Bayesian network [20] is a probabilistic graphical model that represents a set of random variables $\{\boldsymbol{x}_i, i = 1, \ldots, n\}$ and their conditional dependencies via a directed acyclic graph. Each vertex represents a random variable, which may either be discrete or continuous. Directed edges represent conditional dependencies pointing from parent to child vertex;

consequently the conditional distribution of a vertex $\boldsymbol{x}_i$ is given by a probability function $p(\boldsymbol{x}_i \,|\, \mathcal{P}_{\boldsymbol{x}_i}) = f_i(\boldsymbol{x}_i, \mathcal{P}_{\boldsymbol{x}_i})$, where $\mathcal{P}_{\boldsymbol{x}_i}$ denotes the parents of $\boldsymbol{x}_i$. This is illustrated in Figure 1a. If $\boldsymbol{x}_i$ and $\mathcal{P}_{\boldsymbol{x}_i}$ correspond to discrete random variables, then f_i can be represented by a table with one entry for each possible combination of the parents' and node's states. In the continuous case the conditional distribution is described by a probability density function depending on the node's parents.

Because a Bayesian network is a complete probabilistic model for a set of variables it can be used to perform inference of unobserved variables. By the product rule the joint distribution of all variables in a Bayesian network is

$$p(\{\boldsymbol{x}_i, i = 1, \ldots, n\}) = \prod_{i=1}^{n} p(\boldsymbol{x}_i \,|\, \mathcal{P}_{\boldsymbol{x}_i}). \tag{2}$$

Efficient algorithms for exact inference in Bayesian networks require that the utilized distributions are closed under multiplication, since then the computed posteriors are tractable. While this is always the case for discrete distributions represented by a table, in the continuous case this property only applies to exponential family distributions. Consequently inference in general Bayesian networks requires either the use of sampling algorithms, for example Markov chain Monte Carlo (MCMC) sampling, or the approximation of the true conditionals by normal distributions.

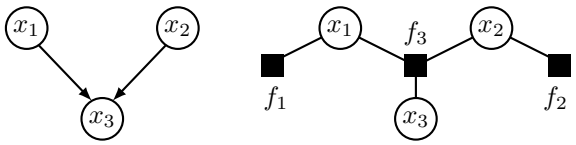
In the context of time series models unscented Kalman filters (UKFs) [14] estimate the shape of the true posterior distribution by evaluating it at so-called sigma points and use the results to approximate the posterior by a best-fit Gaussian. Ko *et al.* [16] successfully employed this approximation to a model with conditional distributions determined by GPs and demonstrated its ability to handle various time series datasets. Deisenroth *et al.* [17] improve upon this approach by replacing the unscented transform with an analytic calculation of the posterior and use moment matching to approximate the latter with a Gaussian.

1) Factor Graphs: Factor graphs [21], [20] are a graphical representation of the factorization of a function and are used as a tool to efficiently evaluate probability functions defined by Bayesian networks. They are undirected, bipartite graphs that have two types of vertices: factors and variables. Consider a function $f(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n)$ that can be decomposed into

$$f(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n) = \prod_{j=1}^{n} f_j(\mathscr{X}_j), \tag{3}$$

where $\mathscr{X}_j \subseteq \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$. The corresponding factor graph contains a variable vertex for each variable $\boldsymbol{x}_i$, and a factor vertex for each function f_j. An edge between factor f_j and variable $\boldsymbol{x}_i$ exists if and only if f_j depends on $\boldsymbol{x}_i$. This is illustrated in Figure 1b.

Since (2) provides an explicit factorization of a Bayesian network into the form of (3), there exists a corresponding factor graph that represents the Bayesian network's joint distribution function, as can be seen from Figure 1.



(a) Bayesian network. (b) Corresponding factor graph.

Fig. 1: Illustration of relationship between Bayesian networks and factor graphs. (a) This Bayesian network represents $p(x_1, x_2, x_3) = p(x_1)p(x_2)p(x_3 \,|\, x_1, x_2)$. (b) The corresponding factor graph with $f_1(x_1) = p(x_1)$, $f_2(x_2) = p(x_2)$, and $f_3(x_1, x_2, x_3) = p(x_3 \,|\, x_1, x_2)$ represents the factorization $f(x_1, x_2, x_3) = f_1(x_1)f_2(x_2)f_3(x_1, x_2, x_3) = p(x_1)p(x_2)p(x_3 \,|\, x_1, x_2)$. Adapted from [20].

2) Inference: the Sum-Product and Max-Sum Algorithms: The sum-product and max-sum algorithms [22], [20] are efficient, exact inference algorithms for Bayesian networks that consist of discrete variables or whose conditional probabilities are given by exponential family distributions. The sum-product algorithm computes the marginal probability $p(\boldsymbol{x}_i)$ of a random variable $\boldsymbol{x}_i$. The max-sum algorithm determines the joint state $\{\tilde{\boldsymbol{x}}_i, i = 1, \ldots, n\}$ with the highest probability $p(\{\tilde{\boldsymbol{x}}_i, i = 1, \ldots, n\})$.

Both algorithms take as input the factor graph corresponding to the joint distribution represented by the Bayesian network, and work by passing messages along the edges of the factor graph according to the following rules: A vertex sends a message along an edge if it has received messages via all other edges. Since a factor graph is bipartite, a message is always passed between a variable and a factor or vice versa. A message is a function, depending on the variable that is represented by the vertex involved in the passing. The messages for the sum-product and max-sum algorithm are shown in Table I. When a factor passes a message to a variable $\boldsymbol{x}_k$ the max-sum algorithm additionally records the states of the variables $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{k-1}, \boldsymbol{x}_{k+1}, \ldots, \boldsymbol{x}_n\}$ that are the winners of the maximum operation. Messages are passed until each vertex has received a message from all its neighbors. After executing the sum-product algorithm the marginal of a variable $\boldsymbol{x}_k$ is given by

$$p(\boldsymbol{x}_k) = \frac{1}{Z} \prod_{j=1}^{m} \mu_{f_j \to x_k}(\boldsymbol{x}_k),$$

where Z is a normalization constant. Similarly, after executing the max-sum algorithm, we have

$$\max_{\boldsymbol{x}_{-k}} p(\boldsymbol{x}_k \,|\, \boldsymbol{x}_{-k}) = \frac{1}{Z} \sum_{j=1}^{m} \exp\big(\mu_{f_j \to x_k}(\boldsymbol{x}_k)\big) \ ,$$

where $\boldsymbol{x}_{-k} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{k-1}, \boldsymbol{x}_{k+1}, \ldots, \boldsymbol{x}_n\}$. Thus the maximum probability is given by

$$p_{\max} = \max_{\boldsymbol{x}_k} \left[\sum_{j=1}^{m} \exp\big(\mu_{f_j \to x_k}(\boldsymbol{x}_k)\big) \right],$$

where x_k is an arbitrarily chosen variable vertex. To compute the joint state $\{\hat{\boldsymbol{x}}_1, \ldots, \hat{\boldsymbol{x}}_n\}$ that has probability $p_{\max} = p(\hat{\boldsymbol{x}}_1, \ldots, \hat{\boldsymbol{x}}_n)$ the variable values producing the maximum are backtracked through the factor graph.

TABLE I: MESSAGES PASSED BY THE MAX-SUM AND SUM-PRODUCT ALGORITHMS. THE SET OF NEIGHBORS OF A NODE v ARE DENOTED BY $\mathrm{ne}(v)$. HERE, $\boldsymbol{x}_{-k}$ IS USED AS A SHORTHAND FOR $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{k-1}, \boldsymbol{x}_{k+1}, \ldots, \boldsymbol{x}_n$.

pair of vertices	passed message
sum-product algorithm:	
factor f_i to variable $\boldsymbol{x}_k$	$\mu_{f_i \to \boldsymbol{x}_k}(\boldsymbol{x}_k) = \sum_{\boldsymbol{x}_{-k}} f_i(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \cdot \prod_{l \in \mathrm{ne}(f_i) \setminus \{k\}} \mu_{x_l \to f_i}(\boldsymbol{x}_l)$
variable $\boldsymbol{x}_k$ to factor f_i	$\mu_{\boldsymbol{x}_k \to f_i}(\boldsymbol{x}_k) = \prod_{j \in \mathrm{ne}(\boldsymbol{x}_k) \setminus \{i\}} \mu_{f_j \to \boldsymbol{x}_k}(\boldsymbol{x}_k)$
leaf factor f_i to variable $\boldsymbol{x}_k$	$\mu_{f_i \to \boldsymbol{x}_k}(\boldsymbol{x}_k) = f_i(\boldsymbol{x}_k)$
leaf variable $\boldsymbol{x}_k$ to factor f_i	$\mu_{\boldsymbol{x}_k \to f_i}(\boldsymbol{x}_k) = 1$
max-sum algorithm:	
factor f_i to variable $\boldsymbol{x}_k$	$\mu_{f_i \to \boldsymbol{x}_k}(\boldsymbol{x}_k) = \max_{\boldsymbol{x}_{-k}} \left[\ln f_i(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) + \sum_{l \in \mathrm{ne}(f_i) \setminus \{k\}} \mu_{\boldsymbol{x}_l \to f_i}(\boldsymbol{x}_l) \right]$
variable $\boldsymbol{x}_k$ to factor f_i	$\mu_{\boldsymbol{x}_k \to f_i}(\boldsymbol{x}_k) = \sum_{j \in \mathrm{ne}(\boldsymbol{x}_k) \setminus \{i\}} \mu_{f_j \to \boldsymbol{x}_k}(\boldsymbol{x}_k)$
leaf factor f_i to variable $\boldsymbol{x}_k$	$\mu_{f_i \to \boldsymbol{x}_k}(\boldsymbol{x}_k) = \ln f_i(\boldsymbol{x}_k)$
leaf variable $\boldsymbol{x}_k$ to factor f_i	$\mu_{\boldsymbol{x}_k \to f_i}(\boldsymbol{x}_k) = 0$

Both algorithms are only guaranteed to produce meaningful results if the factor graph contains no loop, i.e., the Bayesian network must be a polytree. Nevertheless, the sum-product and max-sum algorithms can also be applied to graphs with loops; in this case they produce approximations which converge towards the true posteriors with increasing number of iterations. To initialize message passing dummy messages are injected into the factor graph to break dependency cycles. Then messages are passed according to the rules described above until either convergence is reached or a predetermined set of iterations has elapsed. This method is called *loopy belief propagation* [23] and has been successfully employed for decoding of error correcting codes [24].

III. METHODS

A. Practical Heteroscedastic Gaussian Process Regression

While the heteroscedastic GP model described in Section II-B integrates the varying noise level naturally into the probabilistic framework of the GP, it comes at the cost of an increased computational complexity, since data and noise models must be fitted iteratively, even though approximations are used.

Here we present an approximative method for practical heteroscedastic regression using GPs that requires only twice the training time of a homoscedastic GP. The key idea to avoid iterative optimization is to use an unmodified homoscedastic GP model for regression and adjust the prediction uncertainty via a second GP that has been fitted on the difference between the empirical and the predicted variances. More formally, consider a set of input and output data $\mathcal{D} = \{(\boldsymbol{x}_i, y_i), i = 1, \ldots, n\}$. First a standard, homoscedastic Gaussian process GP_m is trained by maximizing the prediction likelihood on $\mathcal{D}$ (Figure 2a). For each input point $\boldsymbol{x}_i$ the difference between the empirical variance and the variance estimate of GP_m is calculated,

$$z_i = (y_i - \mathrm{GP}_m^{\mu}(\boldsymbol{x}_i, \mathcal{D}))^2 - \mathrm{GP}_m^{\sigma}(\boldsymbol{x}_i, \mathcal{D}),$$

forming a new dataset $\mathcal{D}_v = \{(\boldsymbol{x}_i, z_i), i = 1, \ldots, n\}$. Finally, a second Gaussian process GP_v, with its own covariance function and an independent set of hyper-parameters $\boldsymbol{\theta}_v$, is fitted on $\mathcal{D}_v$ by maximizing the prediction likelihood (Figure 2b). Heteroscedastic regression is performed by combining the predictions of GP_m and GP_v according to

$$\mathrm{HGP}^{\mu}(\boldsymbol{x}^*, \mathcal{D}) = \mathrm{GP}_m^{\mu}(\boldsymbol{x}^*, \mathcal{D}), \tag{4a}$$

$$\mathrm{HGP}^{\sigma}(\boldsymbol{x}^*, \mathcal{D}) = \max(0, \mathrm{GP}_m^{\sigma}(\boldsymbol{x}^*, \mathcal{D}) + \mathrm{GP}_v^{\mu}(\boldsymbol{x}^*, \mathcal{D})). \tag{4b}$$

Results are shown in Figure 2c.

Because the heteroscedastic noise is not incorporated into the probabilistic model of the GP, all samples from $\mathcal{D}$ have the same impact on the predicted mean (4a) irrespective of their noise levels. Thus our approach uses the empirical variance estimation only to refine the prediction uncertainty (4b) over a homoscedastic GP.

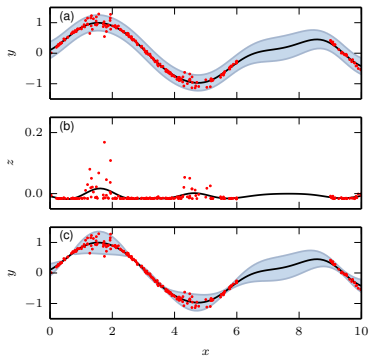


Fig. 2: Practical heteroscedastic GP regression on an artificial dataset. (a) Predictions of a standard, homoscedastic Gaussian process GP_m (black line) and associated standard deviation using the red points as training set. (b) A secondary Gaussian process GP_v (black) has been trained on the difference between the empirical variance and the variance predicted by GP_m (red points). (c) Our heteroscedastic GP correctly predicts the data variance by adjusting the variance predictions of GP_m using GP_v. The high variance at $1 \leq x \leq 2.5$ and $4 \leq x \leq 5$ is due to high training data variance while at $6 \leq x \leq 9$ it is caused by the lack of training points.

B. A Dynamical Probabilistic Model for Sensor Data

We model the sensor's data stream using a Bayesian network. For each time step $t \in \{1, \dots, T\}$ it consists of three random variables: measured quantity q_t, latent state h_t and sensor output $\boldsymbol{x}_t$. Additionally, we model the unknown initial latent state as a random variable h_0. The graphical structure of this time-series model is shown in Figure 3 (the light gray elements are used for smoothing only and should be ignored at this point).

We assume that no prior information about the measured quantity, other than its limits $q_{\min}$ and $q_{\max}$, are available, hence $q_t \sim \mathcal{U}(q_{\min}, q_{\max})$. The conditional state transition probabilities are modeled by a heteroscedastic GP, $h_t \,|\, q_t, h_{t-1} \sim \mathrm{HGP}_t(\{q_t, h_{t-1}\}, \mathcal{D}_t)$. We implemented the HGP as described in Section III-A, however this is not essential and any HGP implementation can be used in this place.

The causal relationship described by our sensor model demands a generative model $p(\boldsymbol{x}_t \,|\, h_t)$ of the observations. Using another GP for this purpose would have two drawbacks: First, GP regression into a d-dimensional output space involves training of d GPs, quickly making the training computationally expensive. Second and more important, application of, e.g., the max-sum algorithm for maximum-likelihood inference of $\{q_t, t = 1, \dots, T\}$ given $\{\boldsymbol{x}_t, t = 1, \dots, T\}$ requires summation of $p(\boldsymbol{x}_t \,|\, h_t)$ over h_t, which would involve the evaluation of the GP for each possible value of h_t.

The key idea to avoid these problems is that Bayes' rule can be used during inference to replace $p(\boldsymbol{x}_t \,|\, h_t)$ by an observational model $p(h_t \,|\, \boldsymbol{x}_t)$ even without having a model for $p(\boldsymbol{x}_t)$. For the observational model, $p(h_t \,|\, \boldsymbol{x}_t)$, a heteroscedastic GP, HGP_h, is used and thus we have

$$p(\boldsymbol{x}_t \,|\, h_t) = \frac{\mathcal{N}(h_t \,|\, \mathrm{HGP}_h^{\mu}(\boldsymbol{x}_t, \mathcal{D}_h), \mathrm{HGP}_h^{\sigma}(\boldsymbol{x}_t, \mathcal{D}_h))\; C(\boldsymbol{x}_t)}{p(h_t)},$$

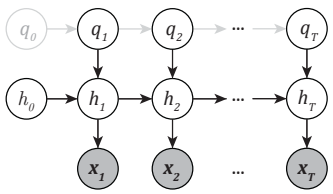
where $C(\boldsymbol{x}_t)$ is constant w.r.t. the other variables. Calculation of $p(h_t)$ in the employed Bayesian network would require marginalization over q_t. However, since q_t is random we approximate $p(h_t)$ with a non-informative distribution, i.e., we assume that $h_t \sim \mathcal{U}(h_{\min}, h_{\max})$ where $h_{\min} \in \mathbb{R}, h_{\max} \in \mathbb{R}$ with $h_{\min} < h_{\max}$. Furthermore, since no information about the latent state of the sensor is available at the beginning of an experiment, we also assume $h_0 \sim \mathcal{U}(h_{\min}, h_{\max})$.

C. Probabilistic Smoothing of q_t

If the measured quantity is expected to be smooth over time, the prediction result can be improved by incorporating this assumption in the Bayesian network. The uniform prior for q_t is replaced by a Gaussian distribution with mean q_{t-1} and variance σ_s^2 that corresponds to the desired level of smoothness. Additionally the random variable $q_0 \sim \mathcal{U}(q_{\min}, q_{\max})$ is introduced to start the chain. The elements that are added to the model are depicted in light gray in Figure 3.

D. Training

A common approach to train models with latent variables is the well-known Expectation Maximization (EM) algorithm, an iterative procedure that estimates a distribution over the latent variables and then uses this estimate to update the model's conditional distributions, which in our case would involve fitting two heteroscedastic GPs per iteration. Since many iterations might be necessary to reach convergence the algorithm is computationally expensive.



probability	distribution function
common:	
$p(h_0)$	$\mathcal{U}(h_0 \,\|\, 0, s_{\max})$
$p(h_t \,\|\, q_t, h_{t-1})$	$\mathcal{N}\big(h_t \,\|\, \mathrm{HGP}_t^{\mu}(\{q_t, h_{t-1}\}, \mathcal{D}_t),$ $\mathrm{HGP}_t^{\sigma}(\{q_t, h_{t-1}\}, \mathcal{D}_t)\big)$
$p(\boldsymbol{x}_t \,\|\, h_t)$	$\mathcal{N}\big(h_t \,\|\, \mathrm{HGP}_h^{\mu}(\boldsymbol{x}_t, \mathcal{D}_h), \mathrm{HGP}_h^{\sigma}(\boldsymbol{x}_t, \mathcal{D}_h)\big) \cdot$ $C(\boldsymbol{x}_t) \,/\, p(h_t)$
non-smoothing:	
$p(f_t)$	$\mathcal{U}(f_t \,\|\, 0, f_{\max})$
smoothing:	
$p(f_0)$	$\mathcal{U}(f_0 \,\|\, 0, f_{\max})$
$p(f_t \,\|\, f_{t-1})$	$\mathcal{N}(f_t \,\|\, f_{t-1}, \sigma_s^2)$

Fig. 3: Our model for dynamic sensor data processing is shown as a Bayesian network with associated distribution functions. For each time step $t \in \{1 \dots, T\}$, q_t corresponds to the quantity that is measured by the sensor, h_t to the internal state and $\boldsymbol{x}_t$ to the sensor's output. During inference only $\boldsymbol{x}_t$ is observed. The light gray connections can optionally be included to probabilistically smooth the predicted quantities.

Here we use a different, non-iterative training procedure. The key idea is to train the observational model $p(h_t \,|\, \boldsymbol{x}_t)$ using the quantity q_t as the regression target, but then use the model's predictions as estimates for the latent state h_t. Consider a set of observed time series consisting of the measured quantity and the corresponding output of the sensor, $\mathcal{D}_h = \{(\boldsymbol{x}_t^i, q_t^i), t = 1, \dots, T, i = 1, \dots, n\}$. We optimize the hyper-parameters of HGP_h by maximizing the likelihood of $\mathcal{D}_h$. The trained GP will not be able to predict q_t from $\boldsymbol{x}_t$ with high accuracy since it can only capture instantaneous correlations between quantity and output; however we assume that its predictions correspond to the sensor's internal state h_t. We then set $h_t^i = \mathrm{HGP}_h^{\mu}(\boldsymbol{x}_t^i, \mathcal{D}_h)$ and use the obtained predictions to create the training set $\mathcal{D}_t = \{([q_t^i, h_{t-1}^i], h_t^i), t = 2, \dots, T, i = 1, \dots, n\}$. The hyper-parameters of HGP_t are optimized by maximizing the likelihood of $\mathcal{D}_t$.

At this point h_t can be predicted using either HGP_h from $\boldsymbol{x}_t$ or HGP_t from the time series $\{q_t, t = 1, \dots, T\}$. This can be used to validate if our model correctly captured the hysteresis by cross-checking both predictions on a test set.

E. Inference of q_t

Standard algorithms that perform closed-form inference are not applicable to our model as-is, since the GPs in the conditional probability functions of variables in the Bayesian network (Figure 3) can have arbitrary, non-analytic posterior

distributions. However, in our particular setting, the unobserved variables q_t and h_t are both one-dimensional and hence, after training, we can deal with this problem by discretization of q_t, h_t and their corresponding conditional probabilities using a fixed quantization interval Δs.

If no probabilistic smoothing is desired, the standard max-sum algorithm (Section II-C2) is used to infer the most likely time series $\{q_t, t = 1, \ldots, T\}$ given the sensor's output $\{\boldsymbol{x}_t, t = 1, \ldots, T\}$. Prediction of q_t can be performed as a new $\boldsymbol{x}_t$ arrives without the need to recompute the previous time steps of the Markov chain; thus online inference gives the most likely q_t given $\{\boldsymbol{x}_i, i = 1, \ldots, t\}$. To determine the most likely $\{q_t, t = 1, \ldots, T\}$ given a complete time series $\{\boldsymbol{x}_t, t = 1, \ldots, T\}$, backtracking of the most likely states of h_t and q_t is performed, starting at h_T. Confidence intervals are obtained from the marginal distributions of q_t calculated by the sum-product algorithm.

If probabilistic smoothing is used to improve the accuracy, the additional model elements (Section III-C) cause the factor graph to contain loops which requires the application of approximate inference. Hence we apply the following inference procedure: First the smoothing connections are removed from the factor graph and the standard max-sum message passing algorithm is run until each node has received a message from all its neighboring nodes. Then, with the smoothing connections restored, loopy belief propagation, as described in Section II-C2, is performed. Since at this point the max-sum algorithm already provided reasonable probability estimates for the variables in the factor graph, only few iterations are necessary to smooth the posteriors for q_t between neighboring time steps.

Since the smoothing is done using probabilistic inference it takes full account of the prediction uncertainties of the state evolution and observation GPs. For instance regions in the predicted time series $\{q_t, t = 1, \ldots, T\}$ which have a high certainty according to the model are barely affected by the smoothing. Consequently even sharp steps can be predicted from sensor data; this would not be possible if a conventional low-pass filter would be used to smooth the time series. In contrast, regions in the predicted $\{q_t, t = 1, \ldots, T\}$ with low prediction certainty usually correspond to sensor data with high noise levels and thus a high level of smoothing in these regions is advantageous to filter the noise.

IV. Experiments Using the DLR Artificial Skin

The testbed for our model is the DLR artificial skin, a flexible polymer-based tactile sensor. It consists of a top and bottom layer of circuit tracks made from a non-conductive polymer enriched with carbon black particles, oriented in x- and y-direction, respectively. The tracks are insulated from each other by non-conductive polymer spacer arrays. Each crossing point of the circuit tracks forms a tactile sensing element, called taxel [3]. The sensor is available in several configurations, which differ in thickness, spatial resolution and sensor size [4]. The particular sensor used in this work consists of 64 taxels arranged in an 8×8 grid. Each taxel has a surface area of $9 \times 9\,\mathrm{mm}^2$ and a thickness of approximately $1.5\,\mathrm{mm}$. The sensor is shown in Figure 4a.

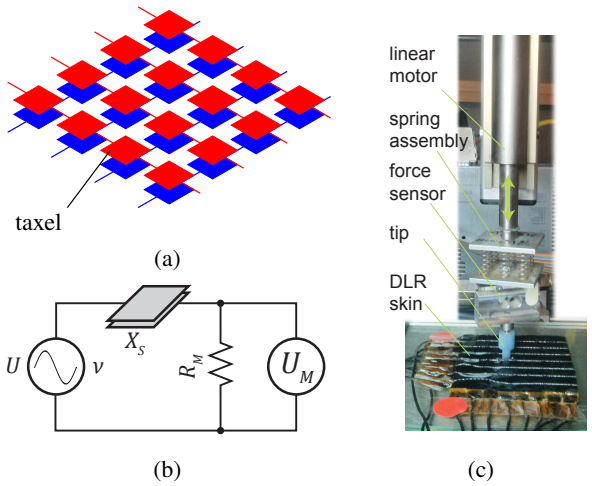


Fig. 4: Skin test bench. (a) Principle of the DLR tactile sensor. The top and bottom layer of a taxel constitute a parallel plate capacitor. (b) Measurement circuit for a taxel of the artificial skin. (c) Mechanical indentor setup. The skin is positioned on an x/y table that allows to position an arbitrary taxel under the indentor.

When pressure is applied to a taxel the polymer is compressed as the thickness of the sensor is reduced locally. This changes the electrical properties (resistance and capacitance) of the polymer, which can be sensed by a measurement circuit connected to the corresponding top and bottom circuit tracks. The level of change depends on the amount of compression. Thus not only the presence of a force, but also its magnitude can be sensed.

A. Test Setup

Figure 4c shows the utilized test bench for stimulating and recording data from the artificial skin. The setup consists of two major components: an indentor assembly that applies controlled pressure onto a taxel, and an x/y table on which the skin is mounted.

The x/y table (Movtec Wacht L60) is used to position a specific taxel under the indentor and has a positioning repeatability of $0.1\,\mathrm{mm}$. The indentor assembly consists of a linear motor, a spring assembly, a force sensor and an indenting tip. The top part of the indentor is driven by a linear motor (LinMot PS02-23Sx80-F / PL01-12x170/120 with controller E1250-EC-UC/V1RE) in position controlled mode with a repeatability of $0.05\,\mathrm{mm}$. Eight springs (Gutekunst Federn VD-090K), assembled in a rectangular $3 \times 3\,\mathrm{cm}^2$ configuration with a resulting spring rate of $k = 7.19\,\mathrm{N/mm}$, are compressed by downward movement of the indentor and thus translate position into force. A force sensor (ME-Systeme KD40s 20N) embedded in the indentor measures the force applied on the skin and converts it into a voltage that passes through an preamplifier (ME-Systeme GSV-11H) with an integrated $20\,\mathrm{Hz}$ low-pass filter (3rd order Bessel) and is digitized by an AD converter (NI WLS-9163 with NI 9205 module). The combined force measurement accuracy of sensor, preamplifier and converter is $0.05\,\mathrm{N}$. The indenting tip consists of a circular, non-conductive, rigid, flat surface with a diameter of $5\,\mathrm{mm}$.

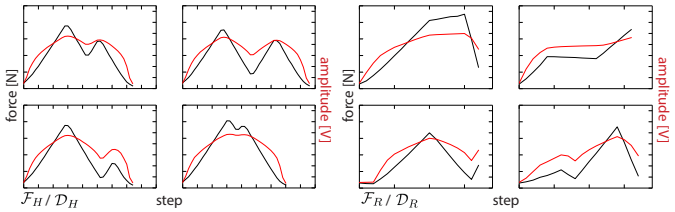


Fig. 5: Samples of force curves (black) and the associated raw skin response (red), i.e., amplitude of the AC voltage at the series resistor R_M. Dataset $\mathcal{D}_H$ (left) consists of 40 curves, of which 10 are used for testing. Dataset $\mathcal{D}_R$ (right) consists of 85 curves, of which 20 are used for testing.

A taxel of the skin is connected in standard voltage-divider configuration (Figure 4b) to a programmable waveform generator (embedded in Agilent DSO-X 2014A). The voltage drop U_M across the series resistor $R_M = 2.7\,\mathrm{k}\Omega$ is recorded and digitized using a 100 MHz oscilloscope (Agilent DSO-X 2014A).

B. Force Stimulation Curves

A force curve $f^i = \{f_t^i, t = 1, \ldots, T\}$ controls the stimulation of a taxel over time. Two sets $\mathcal{F}_R$ and $\mathcal{F}_H$ of force curves, which are generated using different methods, are used. Samples from both force curves are shown in Figure 5.

The purpose of force curve set $\mathcal{F}_H$ is to swipe over the latent state space of the skin effectively and therefore obtain a dataset which captures the taxel's hysteretic behavior. For each curve of set $\mathcal{F}_H$ three forces f_a, f_b, f_c are sampled consecutively from a uniform distribution with the constraint that $f_b < f_c < f_a$. A piecewise linear curve is then generated by starting with zero force, increasing the force with a constant rate $r = \Delta f / \Delta t$ to f_a, decreasing the force to f_b, increasing it again to f_c and finally reducing it back to zero force. This dataset consists of 40 curves, of which 10 are used for testing. On average each curve consists of 37 support points (see Figure 5 (left)).

Force curve set $\mathcal{F}_R$ consists of movements between randomly chosen forces with a limited force change rate $r_{\max}$. For each curve of the set an integer n, controlling the number of curve segments, is sampled from $\mathcal{U}_d(n_{\min}, n_{\max})$ where $0 < n_{\min} < n_{\max}$ and $\mathcal{U}_d$ is the discrete uniform distribution. Then $n-2$ time points $\{t_i, i = 2, \ldots, n-1\}$ are chosen randomly according to $t_i \sim \mathcal{U}_d(1, T)$. We set $t_1 = 0$ and $t_n = T$. For each t_i with $i \in \{2, \ldots, n-1\}$ a force f_{t_i} is sampled according to $f_{t_i} \sim \mathcal{U}(f_{t_i}^{\min}, f_{t_i}^{\max})$ where $f_{t_i}^{\min}$ and $f_{t_i}^{\max}$ are chosen such that $|(f_{t_i} - f_{t_{i-1}})/(t_i - t_{i-1})| \leq r_{\max}$. This limits the maximum force change rate to $r_{\max}$ and thus limits the maximum movement speed of the indentor. The curves are constrained to begin and end with zero force, i.e. $f_1 = f_n = 0$. Finally the force is interpolated linearly between the support points. This dataset consists of 85 curves, of which 20 are used for testing. On average each curve consists of 17 support points (see Figure 5 (right)).

C. Data Recording Prerequisites

Since the used linear motor is position controlled, it cannot be employed to apply a specified force directly. To work around this issue the correspondence between the position of the linear motor and the force measured with the force sensor in the indentor is stored as a force-position mapping $p(f)$. Thus arbitrary forces f can be applied on the artificial skin by moving the linear motor to the corresponding position $p(f)$. Although the precision of the open-loop force control is limited by the indentor assembly, this does not influence the accuracy of the dataset since the actual force is measured by the separate force sensor.

Due to the limited memory of the oscilloscope employed in this preliminary setup it is only possible to record data from the skin in short segments. Thus stimulation curves are executed step-wise and voltage data is recorded while the force is kept stationary.

D. Measurement Procedures

Recording of a force curve f^i is performed as follows: At each step $t \in \{1, \ldots, T\}$ the force f_t^i demanded by the force curve is converted into a position $p_t = p(f_t^i)$ and the linear motor is instructed to drive to this position using constant velocity v_0. The setup awaits that the position is reached and stores the actual force $\hat{f}_t^i$ obtained from the force sensor. Then for each frequency $\nu_j, j = 1, \ldots, m$ from the set of stimulus frequencies $\boldsymbol{\nu}$, the waveform generator is instructed to output a sine wave of frequency ν_j with peak-to-peak amplitude 5 V and no DC offset. The oscilloscope records the resulting voltage curve across the series resistor R_M for approximately 10 periods and transfers it to the control computer. There the discrete Fourier transformation of the signal is computed and the amplitude $a_{t,j}^i$ of the stimulus frequency ν_j is extracted and stored. Thus for each time step we obtain a pair of applied force and corresponding sensor output: $(q_t^i, \boldsymbol{x}_t^i) = (\hat{f}_t^i, [a_{t,1}^i, \ldots, a_{t,m}^i])$.

We now have two datasets of time series: $\mathcal{D}_R = \{\{(q_t^i, \boldsymbol{x}_t^i), t = 1, \ldots, T\}, i = 1, \ldots, n\}$ generated from force curve set $\mathcal{F}_R$, and $\mathcal{D}_H$ corresponding to $\mathcal{F}_H$. Samples from both dataset are shown in Figure 5.

E. Baseline Model

In order to establish a baseline for step-wise prediction of the force q_t from the skin's output $\boldsymbol{x}_t$ we apply standard GP regression using an RBF kernel on the training sets of $\mathcal{D}_R$ and $\mathcal{D}_H$ and evaluate the accuracy on the respective test sets. Regression results for a typical curve from the test set of $\mathcal{D}_H$ are shown in Figure 6a. By design this GP only captures instantaneous correlations between force and voltage and thus hysteresis becomes apparent. The influence of the taxel's internal state, which is not treated in the baseline model, manifests as systematic over- and underestimation of the force.

F. Training

Our model is trained on the training set of $\mathcal{D}_H$ using the procedure described in Section III-D. The discretization step size for q_t is chosen such that the resulting discrete variable has 100 possible states between zero force and the maximum recorded force. The same step size was used to transform the

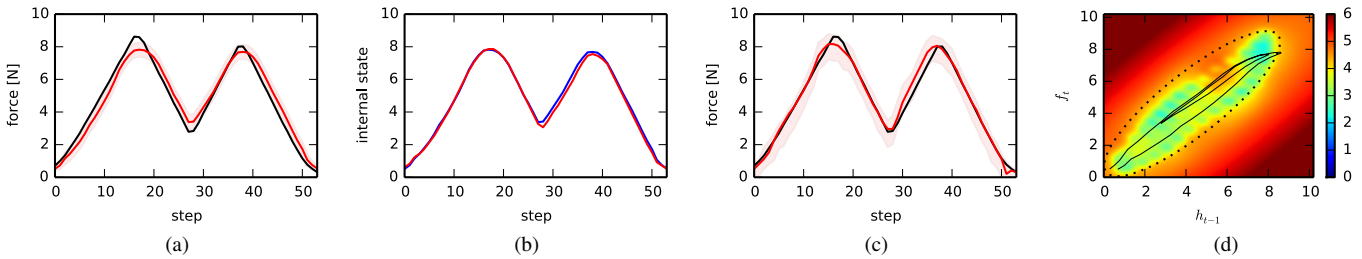


Fig. 6: (a) Baseline force regression without taking the internal state of the skin into account on an exemplary curve from the test set of $\mathcal{D}_H$. Prediction (red) and 68.8% confidence interval (pink) of the applied force (black) were performed using HGP_h. The hysteresis inherent to the taxel's material leads to systematic under- and overestimation. (b) Comparison between internal state prediction from $\boldsymbol{x}_t$ (blue) using HGP_h and from $\{q_1, \ldots, q_t\}$ using HGP_t (red). (c) Force prediction results (red) using our dynamical model with probabilistic smoothing. Hysteresis is significantly reduced compared to (a). Since the inferred distribution of q_t is arbitrary, the confidence interval is not necessarily symmetric around the most likely estimate of q_t. (d) The solid black line shows the predictions of HGP_h vs. the true force. This error plot can be reinterpreted as a phase space plot, where the force is shown on the x-axis and the internal state is plotted on the y-axis. The prediction uncertainty (68.8% confidence interval) of HGP_t after training is color-coded. The available training data $\mathcal{D}_H$ is enclosed by the dotted ellipse. While the average prediction certainty within the region of available training data is high, the area around $(6, 6)$ shows a spike in uncertainty. This may indicate that the Markov property is locally violated, i.e., a one-dimensional h_t cannot represent the internal state of the taxel accurately in this area.

latent state h_t into a discrete random variable. For HGP_h we use the standard RBF kernel and for HGP_t we use the sum of the RBF and linear kernel. To get an intermediate verification of the learned model, we obtain two independent predictions of the internal state h_t using either HGP_t or HGP_h separately on the test set of $\mathcal{D}_H$. Comparing both predictions (Figure 6b) shows that they agree with very high accuracy ($R^2 = 0.99$). Thus we find that for this sensor our non-iterative training method is sufficient to capture hysteresis and a refinement using the EM algorithm is unnecessary. The prediction uncertainty for HGP_t is shown in Figure 6d.

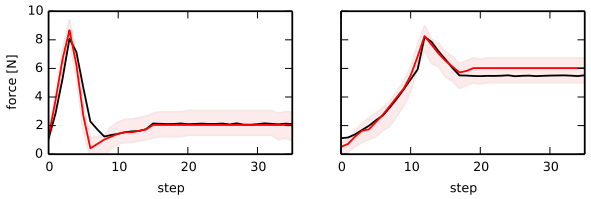


Fig. 7: True force (black) and force prediction results (red) using our dynamical model with probabilistic smoothing on two test force curves with a long period of constant force in the end. The predictions of our model do not drift during the stationary period, although a small offset is sometimes present.

G. Results

Inference of force from the sensor's output was performed according to Section III-E using the test sets of $\mathcal{D}_H$ and $\mathcal{D}_R$. Probabilistic smoothing results use $\sigma_s = 15\,\mathrm{N/step}$. Exemplary regression results are shown in Figure 6c and the accuracy of our dynamical model is summarized in Table II. Furthermore Figure 7 shows that our model is able to accurately predict stationary forces without drifting or oscillating.

Compared to standard, instantaneous GP regression, which does not take temporal dependencies into account, our model halves the average prediction error. In comparison to analytic moment-based GP filtering (GP-ADF) [17], which uses moment-matching to approximate the posteriors by normal distributions in each inference step, our model reduces the prediction error on dataset $\mathcal{D}_H$ by 33%. Since this dataset was designed to capture data with a high amount of hysteresis, it is likely to exhibit multimodal distributions during inference which cannot be approximated properly by a Gaussian.

TABLE II: PREDICTION ACCURACIES MEASURED USING THE ROOT MEAN SQUARED ERROR (RMSE) AND THE COEFFICIENT OF DETERMINATION (R^2). OUR MODEL WAS TRAINED ON THE TRAINING SET OF $\mathcal{D}_H$ AND EVALUATED ON THE TEST SETS OF $\mathcal{D}_H$ AND $\mathcal{D}_R$. THE BASELINE GP REGRESSION USES ONLY THE INSTANTANEOUS $\boldsymbol{x}_t$ TO PREDICT q_t. GP-ADF [17] USES GAUSSIAN APPROXIMATIONS FOR THE POSTERIORS. ONLINE PREDICTION USES $\{\boldsymbol{x}_i, i = 1, \ldots, t\}$ TO INFER q_t WHILE OFFLINE PREDICTION TAKES THE WHOLE TIME SERIES INTO ACCOUNT. THE DATASET $\mathcal{D}_H$, WHICH HAS CAPTURED DATA WITH A HIGH AMOUNT OF HYSTERESIS, SHOWS THE PARTICULAR STRENGTH OF OUR MODEL.

	$\mathcal{D}_H$		$\mathcal{D}_R$	
	RMSE	R^2	RMSE	R^2
GP regression	0.316	0.961	0.224	0.984
GP-ADF				
online	0.261	0.974	0.128	0.995
offline	0.243	0.977	0.123	0.995
proposed dynamical model				
online	0.247	0.976	0.126	0.995
online with prob. smoothing	0.219	0.981	0.131	0.994
offline	0.203	0.984	0.124	0.995
offline with prob. smoothing	0.164	0.990	0.123	0.995

V. CONCLUSION

We have demonstrated that a probabilistic, non-parametric latent variable model can be successfully employed to quantify and compensate for hysteresis of a sensor. While the well-established Preisach model uses a high-dimensional latent space of deterministic binary hysterons, we have modeled

transitions in the system's phase-space *directly* using heteroscedastic GP regression. Consequently our model can perform hysteresis compensation for at least any system for which the Preisach model applies, while additionally providing a confidence interval on the measured quantity. However, since our model is more general than the specialized Preisach model, we require more training data.

For modeling the DLR artificial skin it was sufficient to use a scalar internal state variable. Due to its Markov property the model assumes that the behavior of the system is uniquely determined by the position in phase space, i.e. curves in phase space do not intersect, which corresponds to hysteresis with a local memory. The proposed model can be extended to capture hysteresis with non-local memory by using a multi-dimensional internal state, which stores and propagates relevant information from preceding time steps. However, the dimensionality of the internal state and thus the maximum number of reversions points must be specified in advance. Furthermore, the use of the EM algorithm for internal state inference becomes mandatory.

In this article, we made four contributions to the use of dynamical models for sensor data processing: We introduced a practical method to train heteroscedastic GPs that is fast enough to be used online in a robotic system. We avoided approximation errors by retaining the exact predictive distributions for each time step during inference. In contrast to other dynamical time series models, which use a generative model for the observations given the hidden state, we model the latent state given the observations. This approach has two advantages: GPs are more efficient at mapping data from a high-dimensional space into a low-dimensional space than vice versa; thus in our case training and inference times are shortened. Furthermore the time-intensive application of the EM algorithm is avoided by using the measured quantity as the training target.

VI. Acknowledgments

This project was funded in part by the German Research Foundation (DFG) SPP 1527 Autonomes Lernen and by the TACMAN project, EC Grant agreement no. 610967, within the FP7 framework programme. We thank J. Bayer for his help with preparation of this manuscript and the reviewers for their helpful comments.

References

[1] J. Dargahi and S. Jajarian, "Human tactile perception as a standard for artificial tactile sensing—a review," *The International Journal of Medical Robotics and Computer Assisted Surgery*, vol. 1, no. 1, pp. 23–35, 2004.

[2] R. Dahiya, G. Metta, M. Valle, and G. Sandini, "Tactile sensing—from humans to humanoids," *IEEE Transactions on Robotics*, vol. 26, no. 1, pp. 1–20, 2010.

[3] M. Strohmayr, H. Worn, and G. Hirzinger, "The DLR artificial skin step i: Uniting sensitivity and collision tolerance," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pp. 1012–1018, IEEE, 2013.

[4] M. Strohmayr and D. Schneider, "The DLR artificial skin step ii: Scalability as a prerequisite for whole-body covers," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pp. 4721–4728, IEEE, 2013.

[5] R. A. Shanks and I. Kong, "General purpose elastomers: Structure, chemistry, physics and performance," in *Advances in Elastomers I*, pp. 11–45, Springer, 2013.

[6] B. Persson, "On the theory of rubber friction," *Surface Science*, vol. 401, no. 3, pp. 445 – 454, 1998.

[7] G. Bertotti and I. D. Mayergoyz, *The Science of Hysteresis: 3-volume set*. Academic Press, 2006.

[8] D. Davino, C. Visone, C. Ambrosino, S. Campopiano, A. Cusano, and A. Cutolo, "Compensation of hysteresis in magnetic field sensors employing fiber bragg grating and magneto-elastic materials," *Sensors and Actuators A: Physical*, vol. 147, no. 1, pp. 127 – 136, 2008.

[9] K. Oppermann, B. Arminger, and B. Zagar, "Smart hysteresis compensation of a magneto-elastic force sensor based on terfenol-d," in *Instrumentation and Measurement Technology Conference (I2MTC), 2010 IEEE*, pp. 662–667, May 2010.

[10] C. Visone, "Hysteresis modelling and compensation for smart sensors and actuators," *Journal of Physics: Conference Series*, vol. 138, no. 1, p. 012028, 2008.

[11] C. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[12] G. Welch and G. Bishop, "An introduction to the Kalman filter," tech. rep., University of North Carolina at Chapel Hill, Chapel Hill, NC, USA, 1995.

[13] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Journal of Basic Engineering*, vol. 82, p. 35, Mar. 1960.

[14] E. A. Wan and R. Van Der Merwe, "The unscented Kalman filter for nonlinear estimation," in *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*, pp. 153–158, IEEE, 2000.

[15] J. Ko, D. J. Klein, D. Fox, and D. Haehnel, "Gaussian Processes and Reinforcement Learning for Identification and Control of an Autonomous Blimp," *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pp. 742–747, Apr. 2007.

[16] J. Ko and D. Fox, "GP-BayesFilters: Bayesian filtering using Gaussian process prediction and observation models," *Autonomous Robots*, vol. 27, pp. 75–90, May 2009.

[17] M. P. Deisenroth, M. F. Huber, and U. D. Hanebeck, "Analytic moment-based Gaussian process filtering," in *Proceedings of the 26th annual international conference on machine learning*, pp. 225–232, ACM, 2009.

[18] P. W. Goldberg, C. K. Williams, and C. M. Bishop, "Regression with input-dependent noise: A Gaussian process treatment," *Advances in neural information processing systems*, vol. 10, pp. 493–499, 1997.

[19] K. Kersting, C. Plagemann, P. Pfaff, and W. Burgard, "Most likely heteroscedastic Gaussian process regression," in *Proceedings of the 24th international conference on Machine learning*, pp. 393–400, ACM, 2007.

[20] C. M. Bishop, *Pattern Recognition and Machine Learning*. springer New York, 2006.

[21] J. Pearl, *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 1988.

[22] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *Information Theory, IEEE Transactions on*, vol. 47, no. 2, pp. 498–519, 2001.

[23] K. P. Murphy, Y. Weiss, and M. I. Jordan, "Loopy belief propagation for approximate inference: An empirical study," in *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pp. 467–475, Morgan Kaufmann Publishers Inc., 1999.

[24] R. J. McEliece, D. J. C. MacKay, and J.-F. Cheng, "Turbo decoding as an instance of Pearl's belief propagation algorithm," *Selected Areas in Communications, IEEE Journal on*, vol. 16, no. 2, pp. 140–152, 1998.