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 \( x_t \) 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 \textit{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 \( x_t \) depend on the history of \( q_t \); that is, \( x_t = f_t(q_1, q_2, \ldots, q_t) \). 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 \( x_t = f(h_t) \) for a suitable measurement function \( f \). 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
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 \( D = \{ (x_1, y_1), \ldots, (x_n, y_n) \} \). 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 \( y_i = f(x_i) + \epsilon \), where \( \epsilon \sim \mathcal{N}(0, \sigma^2_i) \).

The kernel function \( k(\cdot, \cdot) \) is a measure of similarity between two inputs. A popular choice is the squared exponential kernel given by

\[
k(x_i, x_j) = \sigma^2 \exp \left( -\frac{\| x_i - x_j \|^2}{2L} \right),
\]

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

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

\[
p(y^* | x^*, D) = \mathcal{N}(GP^\mu(x^*, D), GP^\sigma(x^*, D)),
\]

with mean

\[
GP^\mu(x^*, D) = k^*^T [K_X + R]^{-1} y
\]

and variance

\[
GP^\sigma(x^*, D) = k^* + k^*^T [K_X + R]^{-1} k^* - k^*^T [K_X + R]^{-1} k^T k \cdot [K_X + R]^{-1},
\]

where \( k = [k(x_i, x_1), \ldots, k(x_i, x_n)] \) and \( R = \text{diag}(r^*) \).

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 \( \sigma^2_i \) 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 \( z(\cdot) = \log r(\cdot) \). This z-process uses a different set of hyper-parameters \( \theta_z \) and a separate covariance function \( k_z \). The noise levels \( r \) and \( r^* \) are now described by a probabilistic model and thus the predictions of the heteroscedastic GP can be calculated by marginalization, thus

\[
p(y^* | x^*, D) = \int p(y^* | x^*, z, z^*, D) p(z, z^* | x^*, X) \, dz \, dz^*.
\]

Since analytic calculation of this integral is intractable, Goldberg et al. proposed to obtain Monte Carlo samples \( \{ (z_1, z_1^*), \ldots, (z_k, z_k^*) \} \) from \( p(z, z^* | x^*, X) \) and approximate the integral by

\[
\frac{1}{k} \sum_{j=1}^k p(y^* | x^*_j, z_j, z_j^*, D).
\]

To avoid the time-intensive sampling procedure, Kersting et al. [19] proposed to further approximate \( p(y^* | x^*, D) \) by replacing the marginalization (1) with the most likely point estimate \( (\hat{z}, \hat{z}^*) \). The predictive distribution is thus given by

\[
p(y^* | x^*, D) = p(y^* | x^*, \hat{z}, \hat{z}^*, D).
\]

This approximation is good when most of the probability mass of \( p(z, z^* | x^*, X) \) is located around \( (\hat{z}, \hat{z}^*) \). 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 \( \{x_i, i = 1, \ldots, n\} \) 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 \( x_i \) is given by a probability function \( p(x_i | \mathcal{P}_{x_i}) = f_i(x_i, \mathcal{P}_{x_i}) \), where \( \mathcal{P}_{x_i} \) denotes the parents of \( x_i \). This is illustrated in Figure 1a. If \( x_i \) and \( \mathcal{P}_{x_i} \) 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

\[
p(\{x_i, i = 1, \ldots, n\}) = \prod_{i=1}^{n} p(x_i | \mathcal{P}_{x_i}). \tag{2}
\]

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 \( f(x_1, x_2, \ldots, x_n) \) that can be decomposed into

\[
f(x_1, x_2, \ldots, x_n) = \prod_{j=1}^{n} f_j(\mathcal{X}_j), \tag{3}
\]

where \( \mathcal{X}_j \subseteq \{x_1, x_2, \ldots, x_n\} \). The corresponding factor graph contains a variable vertex for each variable \( x_i \), and a factor vertex for each function \( f_j \). An edge between factor \( f_j \) and variable \( x_i \) exists if and only if \( f_j \) depends on \( x_i \). 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.
TABLE I: Messages passed by the max-sum and sum-product algorithms. The set of neighbors of a node $v$ are denoted by $\text{ne}(v)$. Here, $x_{-k}$ is used as a shorthand for $x_1, \ldots, x_{k-1}, x_{k+1}, \ldots, x_n$.

<table>
<thead>
<tr>
<th>pair of vertices</th>
<th>passed message</th>
</tr>
</thead>
<tbody>
<tr>
<td>sum-product algorithm:</td>
<td></td>
</tr>
<tr>
<td>factor $f_i$ to variable $x_k$</td>
<td>$\mu_{f_i \to x_k}(x_k) = \sum_{x_{-k}} f_i(x_1, \ldots, x_n) \cdot \prod_{l \in \text{ne}(f_i) \setminus {k}} \mu_{x_l \to f_i}(x_l)$</td>
</tr>
<tr>
<td>variable $x_k$ to factor $f_i$</td>
<td>$\mu_{x_k \to f_i}(x_k) = \prod_{j \in \text{ne}(x_k) \setminus {i}} \mu_{x_k \to f_i}(x_k)$</td>
</tr>
<tr>
<td>leaf factor $f_i$ to variable $x_k$</td>
<td>$\mu_{f_i \to x_k}(x_k) = f_i(x_k)$</td>
</tr>
<tr>
<td>leaf variable $x_k$ to factor $f_i$</td>
<td>$\mu_{x_k \to f_i}(x_k) = 1$</td>
</tr>
<tr>
<td>max-sum algorithm:</td>
<td></td>
</tr>
<tr>
<td>factor $f_i$ to variable $x_k$</td>
<td>$\mu_{f_i \to x_k}(x_k) = \max_{x_{-k}} \left[ \ln f_i(x_1, \ldots, x_n) + \sum_{l \in \text{ne}(f_i) \setminus {k}} \mu_{x_l \to f_i}(x_l) \right]$</td>
</tr>
<tr>
<td>variable $x_k$ to factor $f_i$</td>
<td>$\mu_{x_k \to f_i}(x_k) = \sum_{j \in \text{ne}(x_k) \setminus {i}} \mu_{x_k \to f_i}(x_k)$</td>
</tr>
<tr>
<td>leaf factor $f_i$ to variable $x_k$</td>
<td>$\mu_{f_i \to x_k}(x_k) = \ln f_i(x_k)$</td>
</tr>
<tr>
<td>leaf variable $x_k$ to factor $f_i$</td>
<td>$\mu_{x_k \to f_i}(x_k) = 0$</td>
</tr>
</tbody>
</table>

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 $D = \{(x_i, y_i), i = 1, \ldots, n\}$. First a standard, homoscedastic Gaussian process $GP_m$ is trained by maximizing the prediction likelihood on $D$ (Figure 2a). For each input point $x_i$ the difference between the empirical variance and the variance estimate of $GP_m$ is calculated,

$$z_i = (y_i - \text{GP}_m^\mu(x_i, D))^2 - \text{GP}_m^\sigma(x_i, D),$$

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

$$\text{HGP}^\mu(x^*, D) = \text{GP}_m^\mu(x^*, D),$$

$$\text{HGP}^\sigma(x^*, D) = \max(0, \text{GP}_m^\sigma(x^*, D) + \text{GP}_v^\sigma(x^*, D)).$$

Results are shown in Figure 2c.

Because the heteroscedastic noise is not incorporated into the probabilistic model of the GP, all samples from $D$ 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.
B. A Dynamical Probabilistic Model for Sensor Data

We model the sensor’s data stream using a Bayesian network. For each time step \( t \in \{1, \ldots, T\} \) it consists of three random variables: measured quantity \( q_t \), latent state \( h_t \) and sensor output \( x_t \). 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 \( q_{\text{min}} \) and \( q_{\text{max}} \) are available, hence \( q_t \sim \mathcal{U}(q_{\text{min}}, q_{\text{max}}) \). The conditional state transition probabilities are modeled by a heteroscedastic GP, \( h_t \mid q_t, h_{t-1} \sim \text{HGP}_t(\{q_t, h_{t-1}\}, \mathcal{D}_t) \). 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 \( p(x_t \mid h_t) \) 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 is often expensive. Second and more important, application of, e.g., the max-sum algorithm for maximum-likelihood inference is computationally expensive.

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

\[
p(x_t \mid h_t) = \frac{\mathcal{N}(h_t \mid \text{HGP}_t^\mu(x_t, \mathcal{D}_h), \text{HGP}_t^\sigma(x_t, \mathcal{D}_h)) C(x_t)}{p(h_t)},
\]

where \( C(x_t) \) 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 \( h_t \sim \mathcal{U}(h_{\text{min}}, h_{\text{max}}) \) where \( h_{\text{min}} \in \mathbb{R}, h_{\text{max}} \in \mathbb{R} \) with \( h_{\text{min}} < h_{\text{max}} \). Furthermore, since no information about the latent state of the sensor is available at the beginning of an experiment, we also assume \( h_0 \sim \mathcal{U}(h_{\text{min}}, h_{\text{max}}) \).

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 \( q_0 \) is replaced by a Gaussian distribution with mean \( q_{t-1} \) and variance \( \sigma^2_\text{Q} \) that corresponds to the desired level of smoothness. Additionally the random variable \( q_0 \sim \mathcal{U}(q_{\text{min}}, q_{\text{max}}) \) 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 fit two heteroscedastic GPs per iteration. Since many iterations might be necessary to reach convergence the algorithm is computationally expensive.

Here we use a different, non-iterative training procedure. The key idea is to train the observational model \( p(h_t \mid x_t) \) 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, \( \mathcal{D}_h = \{(x_i, q_i^t), t = 1, \ldots, T, i = 1, \ldots, n\} \). We optimize the hyper-parameters of \( \text{HGP}_t \) by maximizing the likelihood of \( \mathcal{D}_h \). The trained GP will not be able to predict \( q_t \) from \( x_t \) 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 \( h_t^1 = \text{HGP}_t^\mu(x_t^1, \mathcal{D}_h) \) and use the obtained predictions to create the training set \( \mathcal{D}_t = \{(q_i^t, h_{t-1}^i, h_t^i), t = 2, \ldots, T, i = 1, \ldots, n\} \). The hyper-parameters of \( \text{HGP}_t \) are optimized by minimizing the likelihood of \( \mathcal{D}_t \).

At this point \( h_t \) can be predicted using either \( \text{HGP}_t \) from \( x_t \) or \( \text{HGP}_t \) from the time series \( \{q_t, t = 1, \ldots, T\} \). 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 \( \Delta s \).

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 \( \{x_t, t = 1, \ldots, T\} \). Prediction of \( q_t \) can be performed as a new \( x_t \) arrives without the need to recompute the previous time steps of the Markov chain; thus online inference gives the most likely \( q_t \) given \( \{x_i, i = 1, \ldots, t\} \). To determine the most likely \( \{q_t, t = 1, \ldots, T\} \) given a complete time series \( \{x_t, t = 1, \ldots, T\} \), 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 \( 8 \times 8 \) grid. Each taxel has a surface area of \( 9 \times 9 \) mm\(^2\) and a thickness of approximately 1.5 mm. The sensor is shown in Figure 4a.

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 indenter 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 indenter and has a positioning repeatability of 0.1 mm. The indenter assembly consists of a linear motor, a spring assembly, a force sensor and an indenting tip. The top part of the indenter 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 mm. Eight springs (Gutekunst Federn VD-090K), assembled in a rectangular \( 3 \times 3 \) cm\(^2\) configuration with a resulting spring rate of \( k = 7.19 \text{N/mm} \), are compressed by downward movement of the indenter and thus translate position into force. A force sensor (ME-Systeme KD40s 20N) embedded in the indenter 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 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 N. The indenting tip consists of a circular, non-conductive, rigid, flat surface with a diameter of 5 mm.
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 \( R_M = 2.7k\Omega \) is recorded and digitized using a 100 MHz oscilloscope (Agilent DSO-X 2014A).

### B. Force Stimulation Curves

A force curve \( f^i = \{f^i_1, t = 1, \ldots, T\} \) controls the stimulation of a taxel over time. Two sets \( F_R \) and \( F_H \) 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 \( F_H \) 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 \( F_H \) three forces \( f_a, f_c, f_b \) 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 \( f_c \), increasing it again to \( f_b \) 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 \( F_R \) consists of movements between randomly chosen forces with a limited force change rate \( r_{\text{max}} \). For each curve of the set an integer \( n \), controlling the number of curve segments, is sampled from \( \mathcal{U}_d(n_{\text{min}}, n_{\text{max}}) \) where \( 0 < n_{\text{min}} < n_{\text{max}} \) and \( \mathcal{U}_d \) is the discrete uniform distribution. Then \( n - 2 \) time points \( \{t_1, t = 2, \ldots, n - 1\} \) 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 \( f_{t_i} \sim \mathcal{U}(f_{t_i_{\text{min}}}, f_{t_i_{\text{max}}}) \) where \( f_{t_i_{\text{min}}} \) and \( f_{t_i_{\text{max}}} \) are chosen such that \( (f_{t_i} - f_{t_{i-1}}) / (t_i - t_{i-1}) \leq r_{\text{max}} \). This limits the maximum force change rate to \( r_{\text{max}} \) 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 \( f^i_t \) demanded by the force curve is converted into a position \( p_t = p(f^i_t) \) 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 \( f^i_t \) obtained from the force sensor. Then for each frequency \( \nu_j, j = 1, \ldots, m \) from the set of stimulus frequencies \( \nu \), the waveform generator is instructed to output a sine wave of frequency \( \nu_j \) with peak-to-peak amplitude 5V 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 \( a_{t,j} \) of the stimulus frequency \( \nu_j \) is extracted and stored. Thus for each time step \( t \) we obtain a pair of applied force and corresponding sensor output: \( (q^i_t, x^i_t) = (f^i_t, a_{t,1}, \ldots, a_{t,m}) \).

We now have two datasets of time series: \( D_R = \{(q^i_t, x^i_t), t = 1, \ldots, T; i = 1, \ldots, n\} \) generated from force curve set \( F_R \), and \( D_H \) corresponding to \( F_H \). 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 \( x_t \) we apply standard GP regression using an RBF kernel on the training sets of \( D_R \) and \( D_H \) and evaluate the accuracy on the respective test sets. Regression results for a typical curve from the test set of \( D_H \) 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 \( D_H \) 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
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 $D_H$. 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.

G. Results

Inference of force from the sensor’s output was performed according to Section III-E using the test sets of $D_H$ and $D_R$. Probabilistic smoothing results use $\sigma_s = 15$ 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 $D_H$ 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>
<thead>
<tr>
<th></th>
<th>$D_H$</th>
<th>$D_R$</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMSE</td>
<td>0.316</td>
<td>0.224</td>
</tr>
<tr>
<td>$R^2$</td>
<td>0.961</td>
<td>0.984</td>
</tr>
</tbody>
</table>

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