Online Spatio-Temporal Gaussian Process Experts with Application to Tactile Classification

Harold Soh
Imperial College London
Email: haroldsoh@imperial.ac.uk

Yanyu Su
Imperial College London
Email: yanyu.su@imperial.ac.uk

Yiannis Demiris
Imperial College London
Email: y.demiris@imperial.ac.uk

***Abstract*—In this work, we are primarily concerned with robotic systems that learn online and continuously from multi-variate data-streams. Our first contribution is a new recursive kernel, which we have integrated into a sparse Gaussian Process to yield the Spatio-Temporal Online Recursive Kernel Gaussian Process (STORK-GP). This algorithm iteratively learns from time-series, providing both predictions and uncertainty estimates. Experiments on benchmarks demonstrate that our method achieves high accuracies relative to state-of-the-art methods. Second, we contribute an online tactile classifier which uses an array of STORK-GP experts. In contrast to existing work, our classifier is capable of learning new objects as they are presented, improving itself over time. We show that our approach yields results comparable to highly-optimised offline classification methods. Moreover, we conducted experiments with human subjects in a similar online setting with true-label feedback and present the insights gained.**

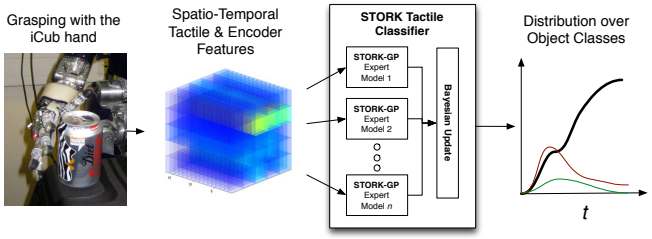


Fig. 1. Our Online Tactile Classifier using STORK-GP Online Learning Experts. Our method works directly on temporal data (without the need for an extensive feature vector) and is capable of creating new spatio-temporal experts "on-the-fly" as new objects are encountered and refining models of familiar objects. A probability distribution over object classes is maintained and updated throughout the grasping action.

I. Introduction

One attribute of successful biological systems is the ability to adapt to changing physical and environmental conditions. This capacity can be exhibited by a short life-cycle with rapid mutations, or in the case of the human species, the continuous development of skills and knowledge throughout life. We believe that robotic systems acting in similarly unconstrained environments will require such *lifelong-learning* characteristics.

In this work, we consider the specific problem of iteratively learning from multiple sensory spatial-temporal data streams. For example, current touch-sensors provide tactile feedback consisting of multi-variate time-varying signals. To be successful at manipulating objects (both familiar and novel), robots are expected to learn from such sensory input in an online-manner to achieve better performance over time.

Our first contribution is a spatio-temporal method that is capable of learning from *multi-variate* data-streams in *online* settings and providing not only predictions but also *uncertainty estimates*. We achieve this by introducing a new recursive kernel based on automatic relevance detection (ARD) [1], integrated into an online Gaussian process (GP) to yield the *Spatio-Temporal Online Recursive Kernel Gaussian Process* (STORK-GP). Experimental results on benchmark problems (i.e., the Mackey-Glass, Henon and Lorenz dynamical systems) demonstrate that STORK-GP performs remarkably well on temporal prediction tasks relative to state-of-the-art online-learning methods.

Encouraged by our results, we applied STORK-GP to the problem of learning to classify objects by touch. Our second contribution is a tactile-based object classifier using STORK-GP as a base "building block" (shown in Fig. 1). In comparison to existing methodologies, our classifier learns *online* as new objects are encountered; it is capable of refining models of existing objects as well as growing new models for novel items. Experiments using the iCub humanoid platform [2] show that our method yields high accuracies, comparable to highly-optimised offline classifiers. Moreover, to better understand how human subjects classify objects by touch with true-label feedback, we conducted experiments with fifteen human-subjects and derived some insights as to the strategies used. Our results add to recent findings of a similar experiment but with the traditional offline-training methodology [3], [4].

The remainder of this paper is organised as follows: in the next section, we present related background material with an emphasis on the Sparse Online Gaussian Process (SOGP) [5], [6]. Section III describes our first contribution: a new recursive kernel with automatic relevance detection that can be used in kernel-based machine learning methods. We then present empirical results on benchmarks in Section IV. Section V details our second contribution: a "growing" generative classifier for tactile-based object classification and our experiments on the iCub humanoid platform. Finally, we conclude with a summary of our findings and a discussion of future work in Section VI.

II. Background

Our method is based on the GP framework and in this section, we give a brief overview of GP regression and a sparse online approximation.

A. Gaussian Processes

Given an observation space with elements $\mathbf{x} \in \mathcal{X}$, a Gaussian process is defined as a "*collection of random variables, any finite number of which have a joint Gaussian distribution*" [7]. It is specified by its mean function,

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})] \tag{1}$$

and its covariance function,

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))].$$

It is often assumed that $m(\mathbf{x}) = 0$ and we write the GP as:

$$f(\mathbf{x}) \sim \mathcal{N}(0, k(\mathbf{x}, \mathbf{x}')). \tag{2}$$

Given a set of identically distributed samples $(\mathbf{x}_i, y_i) \in \mathcal{D}$ where y_i are the associated target values, the *Gram matrix* $\mathbf{K}$ is the matrix of covariances between the $N = |\mathcal{D}|$ training points, i.e. $\mathbf{K} = [k(\mathbf{x}_i, \mathbf{x}_j)]$. Also, let $\mathbf{k}(x') \triangleq [k(\mathbf{x}_i, \mathbf{x}')]_{i=1}^N$. The GP predictive distribution for a new input point $\mathbf{x}_*$ is given by:

$$p(f_*|\mathbf{x}_*, \mathcal{D}) = \mathcal{N}(f_*|\mu_*, \sigma_*^2) \tag{3}$$

where

$$\mu_* = \mathbf{k}(\mathbf{x}_*)^{\mathrm{T}}(\mathbf{K} + \sigma^2\mathbf{I}_N)^{-1}\mathbf{y} \tag{4}$$

and

$$\sigma_*^2 = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}(\mathbf{x}_*)^{\mathrm{T}}(\mathbf{K} + \sigma^2\mathbf{I}_N)^{-1}\mathbf{k}(\mathbf{x}_*). \tag{5}$$

As can be seen from the above discussion, GPs provide *predictive distributions* instead of point predictions. This feature (along with a solid theoretical foundation and good empirical results) has spurred a great deal of interest in GPs over the past decade. In the robotics domain, GPs have been used in a variety of research work (e.g, for learning inverse dynamics [8] and learning-by-demonstration [9]).

B. Online Learning with the Sparse Online Gaussian Process

In this work, we have used the sparse online GP (SOGP) proposed by Csató and Opper [5], [6], an approximation to full GPs that has been used in robotic learning-by-demonstration [10]. In contrast to sparse representations of GPs (see [11] for an overview), SOGP performs fast sequential updates while keeping the model sparse. Detailed descriptions of the SOGP are available elsewhere [5], [6], [12] and as such, we focus on conveying the main intuitions behind the method.

1) The Projected Process Approximation: For online learning, our goal is to update the GP iteratively as observations arrive. Let us denote our observation at time t as $\mathbf{x}^{(t)}$ and our target as $y^{(t)}$. One can update a given GP with a new datapoint observed at $t+1$ by using a Bayesian update [5]. The re-formulated update equations in their "natural parameterisation" forms are given by:

$$m^{(t)}(\mathbf{x}) = \boldsymbol{\alpha}^{(t)\mathrm{T}}\mathbf{k}(\mathbf{x}) \tag{6}$$

$$k^{(t)}(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}, \mathbf{x}') + \mathbf{k}(\mathbf{x})^{\mathrm{T}}\mathbf{C}^{(t)}\mathbf{k}(\mathbf{x}') \tag{7}$$

where $\boldsymbol{\alpha}$ and $\mathbf{C}$ are internally updated (See [5], [6] for details).

2) Maintaining Sparsity: The central idea in limiting the model's growth is to control the number of the datapoints retained, termed "basis vectors" (BVs). We score each incoming point using the following "novelty" score:

$$\gamma(\mathbf{x}^{(t+1)}) = k(\mathbf{x}^{(t+1)}, \mathbf{x}^{(t+1)}) - \mathbf{k}_{\mathcal{B}}^{(t+1)\mathrm{T}}\mathbf{K}_{\mathcal{B}}^{-1}\mathbf{k}_{\mathcal{B}}^{(t+1)} \tag{8}$$

where $\mathbf{k}_{\mathcal{B}}^{(t+1)} = [k(\mathbf{b}_i, \mathbf{x}^{(t+1)})]$ and $\mathbf{K}_{\mathcal{B}}^{-1} = [k(\mathbf{b}_i, \mathbf{b}_j)]$ with $\mathbf{b}_i, \mathbf{b}_j \in \mathcal{B}$. If $\gamma(\mathbf{x}^{(t+1)})$ is below some threshold, ϵ_γ (10^{-4} in our work), then we perform an *approximate* update which absorbs observations but does not increase the size of the BV set, $\mathcal{B}$. Furthermore, to limit the maximum size or *capacity* of $\mathcal{B}$, it may become necessary to delete a basis vector; we score each $\mathbf{b}_i \in \mathcal{B}$ and remove the lowest scoring BV using a reduced update.

The mean of the predictive distribution the SOGP given by:

$$\mu_* = \mathbf{k}_{\mathcal{B}}^{(t)}(\mathbf{x}_*^{(t)})^{\mathrm{T}}\boldsymbol{\alpha}^{(t)} \tag{9}$$

with variance:

$$\sigma_*^2 = k(\mathbf{x}_*^{(t)}, \mathbf{x}_*^{(t)}) + \mathbf{k}_{\mathcal{B}}(\mathbf{x}_*^{(t)})^{\mathrm{T}}\mathbf{C}^{(\mathbf{t})}\mathbf{k}_{\mathcal{B}}^{(t)}(\mathbf{x}_*^{(t)}) \tag{10}$$

Compared to the full GP, the SOGP has a lower computational complexity of $O(s_{\mathcal{B}}^2)$ time where $s_{\mathcal{B}}$ is the maximum BV set size, typically determined by available computational resources.

III. Spatio-Temporal Online Recursive Kernel Gaussian Process (STORK-GP)

In this section, we present the Sparse Online Temporal Recursive Kernel Gaussian Process (STORK-GP); an adapted SOGP with a new recursive kernel formulated for time-series data that enables the specification (or optimisation) of input relevance.

First, we draw attention to the covariance function or *kernel* which plays a significant role in GPs; it defines the notion of closeness or *similarity* between the data points $\mathbf{x}$. For example, the popular squared exponential (SE) kernel has the form:

$$k_{SE}(\mathbf{x}, \mathbf{x}') = \exp\left(\frac{||\mathbf{x} - \mathbf{x}'||^2}{2l^2}\right) \tag{11}$$

where l is the characteristic length scale (a *hyperparameter* of the model). The SE-kernel is isotropic meaning it is invariant to rigid motions (translations and rotations of the entire input space) and thus, encodes our notions of how distance should be measured for the application at hand.

For time-series regression, if the input space is one-dimensional, one can consider a direct application of the SOGP using a sliding-window approach where we construct an "augmented" observational element $\hat{\mathbf{x}}^{(t)} = [x^{(t)}, x^{(t-1)}, \ldots, x^{(t-\tau)}]$ where $x^{(t)}$ is the observation at time t. We can then use the aforementioned SE kernel (or any applicable standard kernel). Although this method can be effective, things become less straight-forward when each data point is multi-dimensional, i.e. $\hat{\mathbf{x}}^{(t)} = [\mathbf{x}^{(t)}, \mathbf{x}^{(t-1)}, \ldots, \mathbf{x}^{(t-\tau)}]$, which is typically the case when dealing with multiple sensors or actuators. It then becomes necessary to vectorise the matrix $\hat{\mathbf{x}}^{(t)}$ and important structural information can be lost in the process. Ideally, we would like the kernel to take into account the temporal nature of sequential observations.

A. Recursive Kernels

Recursive kernels [13] are a recently-proposed class of kernels that share an intimate relationship with recurrent neural networks (RNNs) and were principally derived as a means to extend these networks to infinite size (similar to how the SE kernel discussed earlier can be formulated by considering infinite neural networks). Consider a recurrent network with internal weights $\mathbf{W}$, input weights $\mathbf{V}$ and internal state $\mathbf{s}$. Upon encountering input $\mathbf{x}^{(t)}$ at time t, the RNN output is:

$$\mathbf{y}^{(t)} = h(\mathbf{V}\mathbf{x}^{(t)} + \mathbf{W}\mathbf{s}^{(t)}) \tag{12}$$

where h is a combination of an activation function (such as the hyperbolic tangent) and a projection. The fundamental concept behind the recursive approach is that (12) can be written as:

$$h(\mathbf{W}\mathbf{s}^{(t)} + \mathbf{V}\mathbf{x}^{(t)}) = h\left([\mathbf{W}|\mathbf{V}]\begin{bmatrix}\mathbf{s}^{(t)}\\ \mathbf{x}^{(t)}\end{bmatrix}\right) \tag{13}$$

i.e., a function of the concatenation of the input with the previous state. Realising that the same reasoning could be applied to kernel functions, Hermans and Schrauwen [13] showed that recursive variants of kernels with the form $k(\mathbf{x}, \mathbf{x}') = f(||\mathbf{x} - \mathbf{x}'||^2)$ and $k(\mathbf{x}, \mathbf{x}') = f(\mathbf{x} \cdot \mathbf{x}')$ could be derived in a similar manner[1]. As an example, the recursive-SE kernel has the form: $\kappa^{(t)}(\mathbf{x}, \mathbf{x}') = \exp(-\sigma^{-2}||\mathbf{x}^{(t)} - \mathbf{x}'^{(t)}||^2)$ $\exp\left(\rho^{-2}(\kappa^{(t-1)}(\mathbf{x}, \mathbf{x}') - 1)\right)$. Note that recursive kernels are denoted with the symbol κ to differentiate them from regular kernels.

Using this approach, the authors defined several recursive kernels variants for use in a Support Vector Machine (SVM). Although the recursive kernel can be theoretically applied to time-series of infinite length, in practical settings, it becomes necessary to limit the recursion depth (specified by a parameter τ). In [13], it was demonstrated that the recursive-SE kernel outperformed the standard SE kernel and fixed-sized reservoirs on the NARMA benchmark problem and attained state-of-the-art results on the difficult TIMIT phoneme recognition task.

[1]We refer readers interested in the derivations to [13].

B. Recursive Kernel with Input Relevance

In this work, we propose a recursive kernel based on automatic relevance detection (ARD) [1] which has the form:

$$\kappa^{(t)}_{\text{ARD}}(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{2}(\mathbf{x}^{(t)} - \mathbf{x}'^{(t)})^{\mathrm{T}}\mathbf{M}(\mathbf{x}^{(t)} - \mathbf{x}'^{(t)})\right) \exp\left(\frac{\kappa^{(t-1)}(\mathbf{x}, \mathbf{x}') - 1}{\rho^2}\right) \tag{14}$$

where $\mathbf{M}$ is a symmetric $d \times d$ matrix that controls the *relevancy* of the d inputs. In our experiments, we set $M = \text{diag}(\boldsymbol{l})^{-2}$ where $\boldsymbol{l} = [l_i]_{i=1}^{d}$. Varying the l_i's for different inputs allows us to control the impact that the inputs have on the predictions. From (14), we can see that the kernel function's responsiveness to input dimension k is inversely related to l_k.

From one perspective, $\kappa^{(t)}_{\text{ARD}}$ is a generalisation of both the SE and recursive-SE kernels; if all l_i's are equal, $\kappa^{(t)}_{\text{ARD}}$ reduces to the regular SE recursive kernel. If recursion is not applied, it further reduces to the standard SE kernel. The principal advantage of this generalisation is that it allows for feature weighting/selection while maintaining the intuition that spatial-elements at a time t "belong together". Such input weighting can be difficult to achieve in regular reservoir approaches[2].

It is also worth mentioning that the parameter ρ is related to the spectral radius in echo-state networks [14] and controls the stability of the kernel; typically, the spectral radius is set to less than one. Since there is often a need to optimise the kernel hyper-parameters, we calculated derivatives for $\kappa^{(t)}_{\text{ARD}}$ that can be used for maximising the log marginal likelihood or the pseudo-likelihood:

$$\frac{\partial \kappa^{(t)}_{\text{ARD}}}{\partial l_i} = \kappa^{(t)}_{\text{ARD}}\left[\frac{1}{\rho^2}\frac{\partial \kappa^{(t-1)}_{\text{ARD}}}{\partial l_i} + \frac{\beta_i}{l_i^3}\right] \tag{15}$$

$$\frac{\partial \kappa^{(t)}_{\text{ARD}}}{\partial \rho} = \kappa^{(t)}_{\text{ARD}}\left[\frac{-2(\kappa^{(t-1)}_{\text{ARD}} - 1)}{\rho^3} + \frac{1}{\rho^2}\frac{\partial \kappa^{(t-1)}_{\text{ARD}}}{\partial \rho}\right] \tag{16}$$

where $\beta_i = (x_i^{(t)} - x_i'^{(t)})^2$. These derivatives are also recursive in nature and we have the base cases:

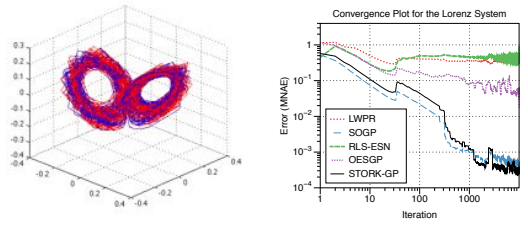
$$\frac{\partial \kappa^{(1)}_{\text{ARD}}}{\partial l_i} = \frac{\beta_i}{l_i^3}\kappa^{(1)}_{\text{ARD}} \quad \text{and} \quad \frac{\partial \kappa^{(1)}_{\text{ARD}}}{\partial \rho} = 0$$

Finally, we note that our proposed recursive kernel can be used with full GPs and other kernel machines. However, we were primarily interested in online learning and hence, integrated it into the SOGP. Implementing this kernel is straight-forward and we have made our code freely available for download [15]. To differentiate our variant from the regular SOGP, we term our method the Spatio-Temporal Online Recursive Kernel GP (STORK-GP).

[2]Typically, the inputs weights are set to 1 are not adapted. Even changing the input weights may help only to a degree since the internal weights that propagate signals in the reservoir are not adapted.

TABLE I
MNAE (TOP) AND RMSE (BOTTOM) SCORES FOR THE ONE-STEP PREDICTION TASK ON BENCHMARKS. IR PROBLEMS HAVE AN ADDITIONAL IRRELEVANT INPUT DIMENSION.

Problem	LWPR	SOGP	RLS-ESN	OESGP	STORK-GP
Mackey-Glass	0.1234	0.1232	0.0053	0.0024	**0.0006**
	0.0314	0.0314	0.0015	0.0007	**0.0002**
Henon	0.2317	0.2200	0.0141	0.0033	**0.0004**
	0.2046	0.1958	0.0130	0.0039	**0.0005**
Lorenz	0.3710	0.0005	0.5033	0.0365	**0.0003**
	0.0494	0.0001	0.0661	0.0081	**0.0002**
Mackey-Glass (IR)	0.1268	0.1235	0.7882	0.8071	**0.0006**
	0.0334	0.0315	0.2023	0.2080	**0.0002**
Henon (IR)	0.2361	0.2380	0.8123	0.7837	**0.0004**
	0.2066	0.2508	0.7006	0.6816	**0.0005**
Lorenz (IR)	0.3844	0.0182	0.8717	0.9016	**0.0003**
	0.0509	0.0050	0.1153	0.1190	**0.0002**



(a) Lorenz Dynamical System (b) Convergence Plot

Fig. 2. Convergence on the Lorenz dynamical system. 2(a) the Lorenz true values (blue) and the tracking performed by STORK-GP (dashed-red). 2(b) shows the (smoothed) testing errors on a sample run for the different algorithms considered (note the log-scale).

C. Relation to Other Methods

The closest related work is the OESGP [12], which combines the echo-state network with SOGP. STORK-GP can be thought of as a extension of OESGP to reservoirs of "infinite size" and in the process, obviating the need to construct a fixed-sized reservoir. This can have computational cost savings (as there is no need to update a potentially large reservoir) and accuracy improvements (described in the following sections). Compared to state-of-the-art online regressors, i.e., locally-weight projection regression (LWPR) [16] and SOGP, the primary difference is that STORK-GP takes into account temporal relationships using the recursive kernel, making it more appropriate when dealing with multivariate data streams.

IV. Empirical Results on Benchmark Problems

In this section, we present empirical results comparing STORK-GP to state-of-the-art methods. We tested STORK-GP on the one-step prediction task using three well-known benchmark problems i.e. the Mackey-Glass, Henon and Lorenz dynamical systems[3]. The objective was to predict the next state $x^{(t+1)}$ given $x^{(t)}$. As performance measures, we used the standard mean normalised absolute error (MNAE) and the root mean squared error (RMSE). For all problems, we set $\tau = 5$ except the Mackey-Glass problem where $\tau = 25$.

[3]MATLAB scripts for generating these time-series are available [17] (which makes use of the Reservoir Computing toolbox [18] and scripts by Wen [19]).

For comparison, we include error scores obtained by LWPR, SOGP, RLS-ESN and OESGP. LWPR results were obtained using the lwpr library with default parameters [20]. For the recursive least squares echo-state network (RLS-ESN) [21] and OESGP, we used fixed-reservoirs with 100 neurons and a spectral radius of 0.99. For these methods, we employed a grid-search on their respective parameters (over reasonable sets) to minimise the error over the first 3000 time-steps. For STORK-GP, we obtained (locally) optimised hyper-parameters by minimising the negative log-likelihood of the full GP via conjugate gradients (using only the initial 500 time-steps).

Our results are shown in Table I. As can be seen, STORK-GP performs remarkably well relative to the other methods, producing accuracies superior even to the recently introduced OESGP. A convergence plot of the different algorithms on the Lorenz problem (Fig. 2) shows the STORK-GP converges quickly and manages to accurately track the relatively complex Lorenz dynamical system. However, STORK-GP's performance increase was balanced by an increased computational cost; it was faster than OESGP (on the order of 10^{-3}s per iteration) but more expensive than RLS-ESN and LWPR (both on the order of 10^{-5}s).

To illustrate the effect of feature selection/weighting, we conducted additional tests whereby we added one extra input dimension consisting of white noise. As can be seen from Tbl. I (problems denoted by IR), STORK-GP retained its accuracy and outperformed the other methods, particularly the online ESN-type algorithms. We observed the optimised characteristic length-scale for the irrelevant dimension was large (on the order 10^5), effectively negating its impact. Encouraged by the promising results for regression-type problems, we used STORK-GP as a "building-block" for our tactile classifier described in the next section.

V. Object Classification by Touch

Our sense of touch is considered our most pervasive sense and is exquisitely sensitive; a large number of nerve receptors distributed across our body allows us to finely examine our environment. Touch sensing allows us to detect, identify and manipulate objects not within our field-of-view; we can easily feel to locate and grasp a pen on a desk behind us while we are attending to some other visual-based task (e.g., reviewing a paper or watching television).

The development of "artificial skin" — tactile sensors created out of flexible semiconducting materials — may provide future robots with similar (or heightened) touch-sensing capabilities. In this section, we contribute an online tactile classifier that uses STORK-GP on the iCub humanoid platform.

A. Related Work

A closely-related work is a recent study by Chitta *et al.* [3], [4] which classified objects based on the high-frequency response using a decision tree. Similarly, we classified objects using tactile feedback during a grasping motion (without lifting the object). However, a primary difference is that our system learns online, i.e., it continuously improves its internal

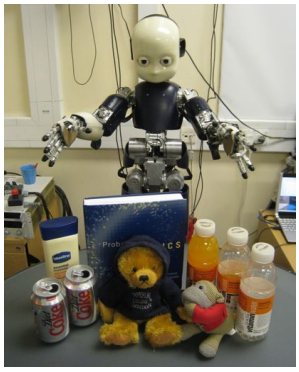


Fig. 3. iCub with Objects: Plastic bottles (full, half-full, empty), Soda cans (empty, half-full), Teddy-bear, Monkey soft-toy, Lotion and Book

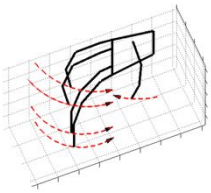


Fig. 4. Illustrated Grasping Motion

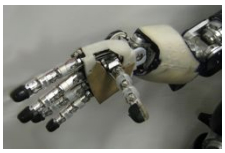


Fig. 5. Grasping Pre-shape

models of familiar things and creates models for novel objects "on-the-fly". We believe such an approach is more applicable to robots operating in the real-world where both novel and familiar items would be frequently encountered. Therefore, instead of using offline classifiers (such as C4.5 decision trees used in [3], [22] or neural networks [23]), we developed an *online generative* model that that does not require the construction of an extensive feature vector (such as the bag-of-features approach used by Schneider *et al.* [24]) and provides a probability distribution over object classes.

Moreover, previous work on tactile classification has focussed mainly on using grippers (e.g., the two-fingered gripper on PR2 robot, [3]) and industrial-style robotic arms [23], [24]. For this study, we used the iCub humanoid robotic platform (Fig. 3) which has two anthropomorphic dexterous hands with 5 fingers (20 joints with 9 degrees of freedom). Each fingertip is wrapped with 12 capacitive pressure sensors under a layer of soft silicone foam. When pressure is applied to the fingertips, the silicone foam is compressed, changing the capacitance. An embedded-board samples all the sensors sequentially, generating an output values ranging from 0 to 255. In this work, the data capture rate was 10Hz.

B. Grasp Controller and Test Objects

We devised a grasp controller that executed a three-step action: first, it fixed the hand position and orientation and then fully opened the hand digits, yielding the pre-shape (as shown in Fig. 5). After we placed an object in a graspable position, the controller closed the digits with low velocity (20 deg/s for each degree of freedom) along a pre-defined trajectory (illustrated in Fig. 4) until the tactile sensor readings breached a critical threshold (or the trajectory completed). In our trials, we found it necessary to vary this threshold from 15-20 to compensate for sensor drift. Finally, the controller would press on the object by moving each digit further along the same trajectory but with a higher velocity (40 deg/s). This continued until the motion was blocked or the trajectory motion was finished. Motion blocks were detected by checking that motor encoder values remained unchanged for 0.5 seconds.

In this experiment, we used nine different everyday objects and one baseline where the grasp was performed with no object, totalling ten classes (shown in Fig. 3). For each of these classes, we created a dataset of twenty samples using our grasp controller and recorded data for the pressing portion of the grasp, yielding a total of 200 samples. We have made this dataset freely available online [25].

C. STORK-GP Tactile Classifier

As can be seen in Fig. 6, each grasped object generates a distinctive spatio-temporal "signature". Our STORK-GP Tactile Classifier (STORK-TC) is based on the notion that each of these signatures can be learned and represented by an "expert model" (Fig. 1).

We used the generative modelling approach whereby each object class $c_i \in \mathcal{C}$ was represented by a separate STORK-GP model, $m_i \in \mathcal{M}$. Our classifier processes the encoder (for each of the 9 DOFs) and tactile sensor data from each finger. Instead of using all twelve sensors directly, the tactile sensor data was reduced by computing the first three moments (mean, standard deviation and skewness) as well as the maximum and minimum reading for the twelve sensors on *each* finger; note that this data reduction is "spatial" and not temporal as we do not compute statistics across the time-steps. This results in an observation/feature vector $\mathbf{o}_t \in \mathbb{R}^{34}$ (25 tactile features and 9 encoder values) for each time step t. We also experimented with *differenced* data streams where only changes in encoder and tactile values are provided to the classifier. Since the initial shape of the fingers is lost, this is a more difficult task but may be more robust against sensor drift.

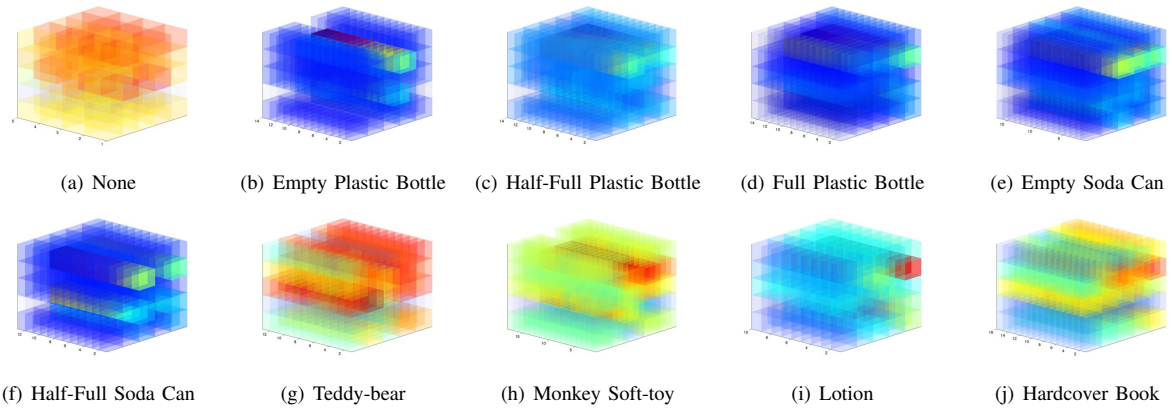
We initialise the system with single model after the first object is encountered and a model class is created whenever a new object is taught to the classifier. In other words, the system "grows" new STORK-GP models as needed. For simplicity, we set each model to use 500 basis vectors with hyper-parameters $\boldsymbol{l} = [100]_{i=1}^{D}$, $\sigma = 0.01$ and $\rho = 0.99$. Inference is performed using a Bayes filter to update a probability distribution over the model classes as observations arrive:

$$p(c_i^{(t)}|\mathbf{o}, \mathcal{M}) = \frac{p(\mathbf{o}^{(t)}|m_i)p(c_i^{(t-1)})}{\sum_j p(\mathbf{o}^{(t)}|m_j)p(c_j^{(t-1)})} \tag{17}$$

where c_t^i is the object class i at time t and $\mathbf{o}$ is the observation (encoder and tactile values). We use a standard observational model where each coordinate was treated independently:

$$p(\mathbf{o}^{(t)}|m_i) = \prod_k N(o_k^{(t)}|y_{i,k}^{(t)}, \sigma_{i,k}^{(t)}) \tag{18}$$

where $y_{i,k}^t$ and $\sigma_{i,k}^{(t)}$ are the k^{th} predicted mean and standard deviations using model m_i. When the grasping motion is initiated, the initial prior is set to be uniformly distributed across the object classes. The class distribution is continuously updated as sensory data is received and the class with the highest probability (i.e., with the model that predicted the



(a) None (b) Empty Plastic Bottle (c) Half-Full Plastic Bottle (d) Full Plastic Bottle (e) Empty Soda Can

(f) Half-Full Soda Can (g) Teddy-bear (h) Monkey Soft-toy (i) Lotion (j) Hardcover Book

Fig. 6. Sample tactile sensor spatio-temporal blocks for the each of the 10 classes where each object generates an individual "signature". Each vertical 5x5 slice represents the 25 tactile features (mean, standard deviation, skewness, maximum and minimum for each finger). Note that these the features change across time (horizontally-stacked slices) as the fingers press on each object.

sensory input best over time) at the end of the grasping action is selected as the recognised object.

In addition to being conceptually straightforward, our approach gives a probability distribution over the different objects during the entire course of the grasping motion. Furthermore, it integrates new information rapidly; once a true-label is offered, the system updates the appropriate object model with the sensory data obtained.

D. Online Learning Experiment

Our online-learning experimental setup is conducted in two phases: in the first phase, a sample from each object class is presented to classifier once[4]. In the second phase, an object is chosen randomly (stratified sampling) and presented for classification. After making a prediction, the algorithm is told the actual class label. As such, the algorithm continuously learns from samples throughout the experiment.

E. Online Classification Accuracy

The results of our experiment (average score across 30 repeated tests with the order of presented samples shuffled) are shown in Tbl. II. As can be seen, STORK-TC achieved near-perfect accuracy on the "normal" data-streams and a lower score on the differenced data (92.3%).

Since class confidence measures were available, we performed a simple thresholding on the predicted outcomes, i.e, we "pass" on low-confidence predictions. At a 99% confidence threshold, STORK-TC attained 100% accuracy while classifying 95% of the objects (on the differenced data, STORK-TC achieved 96% accuracy, passing on 6% of the samples). A mistake that the classifier consistently made was confusing the baseline "none" with the "monkey toy"; a possible reason is suggested by the observation that the small and soft nature of the toy made grasping difficult at times and the hand would close completely.

[4]Strictly speaking, this phase was not necessary since new models can be created as the trials progressed but this simplified comparison with offline methods and the human subjects who knew what the objects were in-advance.

TABLE II
OBJECT CLASSIFICATION ACCURACY. ACCURACY SCORES FOR OFFLINE METHODS ARE THE BEST 5-FOLD CROSS-VALIDATION ACCURACIES.

Methodology	Algorithm	Normal	Differenced
Online	STORK-TC (All)	0.993 (0.002)	0.923 (0.013)
	STORK-TC (Final 20%)	1.000 (0.000)	0.971 (0.033)
Offline	C4.5	0.985 (0.012)	0.935 (0.037)
	SVM	0.835 (0.068)	0.920 (0.043)
	SVM (FS+Opt)	0.995 (0.001)	0.970 (0.034)

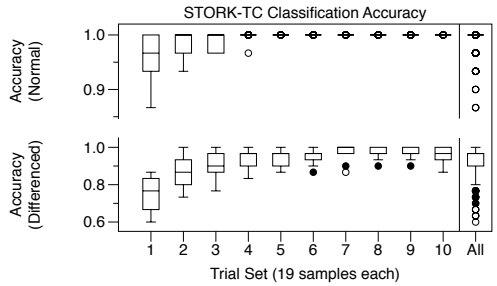


Fig. 7. STORK-TC Classification Accuracy as Trials Progressed. For both the normal and differenced datasets, STORK-TC improves its performance as the trials progressed, as indicated by the higher accuracies and smaller standard deviations. Although the differenced dataset was more difficult (as evidenced by the slower increase in accuracy), by the last twenty percent of the test set (trial sets 9 and 10), STORK-TC achieves 97% (similar to the highly-optimised SVM).

Fig. 7 shows the performance of the algorithm improved as the experiment progressed over time; this can be clearly seen by the higher accuracies and smaller standard deviations across the 30 repeated tests. By the final 20% of the samples, STORK-TC achieved perfect classification accuracy on the regular dataset and 97% on the differenced dataset.

F. Performance Comparison to Offline Classifiers

We were curious how STORK-TC (with the iCub tactile sensors) operating in an online setting would perform relative to offline methods. As such, we compared our method against the C4.5 decision tree (which was used with success in several

TABLE III
FEATURES COMPUTED USED FOR OFFLINE METHODS (COMPUTED FOR EACH FINGER SEPARATELY).

Feature	Description
1-25	First three moments, max and min of the mean tactile data
26-50	First three moments, max and min of the *sd.* tactile data
51-75	First three moments, max and min of the skewness of the tactile data
76-165	Encoder readings during maximum and minimum tactile readings

research works on tactile classification [3], [22]) and SVM using the RapidMiner platform[5]. Since the offline methods do not directly work on temporal data, we created a feature vector similar to [22] by computing statistics across the temporal dimension (shown in Tbl. III). We optimised each algorithm using a grid-search to minimise the 5-fold cross-validation error. Specifically, we optimised the penalty term C and the RBF kernel width for the SVM. We optimised C4.5 parameters (i.e. leaf size, split criteria, minimal gain) in a similar manner.

The best 5-fold cross-validation accuracies of the two methods are shown in Tbl. II. It is important to note that accuracy comparisons should be made *qualitatively* since the accuracy scores are computed differently from the online method. We observed that the decision tree performed well on this problem, better than the SVM by a significant margin. That said, both methods did not obtain the accuracy level attained by STORK-TC on the normal dataset. We hypothesised that the SVM was having difficulty due to the large feature vector (165 features) and we performed an additional feature selection (FS) for the SVM using a genetic algorithm[6] (the objective function was the average 5-fold cross-validation accuracy), followed by an additional optimisation of algorithm parameters. Only after this computationally-heavy optimisation did the SVM attain high scores similar to STORK-TC.

G. Computational Performance

Our computational test-bed was an iMac with a 2.8Ghz Intel Core-i7 processor and 4GB of RAM. On this system, we observed each classification *iteration* (when a new observation was processed) required a check against all experts and thus required more time (0.007s) compared to training a single model (0.0009s). The *total* time taken to classify an object and subsequently train the appropriate model once the true-label was offered was $\approx$ 0.08 seconds (*sd.* 0.04s) — far less than the time needed to execute the grasping motion.

H. Performance Comparison to Human Subjects on Hidden State Determination

In addition to comparisons with offline classifiers, we sought to understand how human subjects would perform given the online-learning experimental setup with true-label feedback after each attempt. From preliminary tests, we discovered that humans easily achieved 100% accuracy when distinguishing the (easily deformed) soda cans, the soft toys (through texture),

[5]http://www.rapid-i.com

[6]We also attempted to use principal component analysis (PCA) for feature reduction but this yielded suboptimal results.

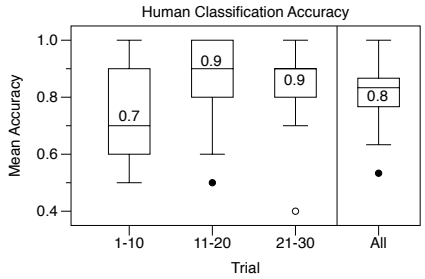


Fig. 8. Human Classification Accuracy as Trials Progressed (with medians shown in the boxes). Note the improvement in accuracy between trials 1-10 and 11-20. Differences between the first and second segments are statistically significant at the 95% level with $p = 0.033$

TABLE IV
CONFUSION MATRIX FOR FULL, HALF-FULL AND EMPTY PLASTIC BOTTLES (HUMAN STUDY)

	Predicted		
True Label	Empty	Half-Full	Full
Empty	111	37	2
Half-Full	17	110	23
Full	0	9	141

the lotion bottle and the hard-cover book. However, they appeared to have difficulty discriminating between between the plastic bottles. Similar findings were reported by Chitta *et al.* [4], where human subjects achieved a classification accuracy score of 75.2% when asked to discriminate between full and empty (both open and closed) plastic bottles — note however that in that study, the humans trained before-hand (until they were confident about their abilities) and were not offered the correct labels during the testing stage.

We invited 15 human subjects (ages 21-38 years, mean 28.5 years, 13 males) to participate in an experiment similar to the classification task described in Section V-D but limited to the three plastic bottles. In the first phase, each participant was allowed to grasp each of the items once and told the object's class; they were not allowed to lift or otherwise move the bottles. In the second phase, they were asked to classify a randomly selected bottle (30 trials with stratified sampling) hidden in covered box. After each attempt, the participants were provided feedback (the true-label), giving them the opportunity to learn from mistakes.

The average human score was 80.4% (*sd.* 12.4%), lower than that achieved by iCub with STORK-TC (which achieved 100% accuracy across 20 repeated tests with the order of test samples shuffled). From the confusion matrix (Tbl. IV), we observed that the full bottle was the easiest to classify. The empty and half-full bottles were more difficult to tell apart; we observed our participants frequently misclassified the empty bottle as half-full, particularly during earlier trials. The one participant who achieved perfect accuracy revealed to us after the experiment that he used a combination of temperature and bottle deformation to help him with the task. Since STORK-TC can easily accommodate other sensor-streams, future work may fuse sensor data to better classify items.

To determine if the subjects improved as the trials progressed, we segmented the data into three portions, i.e., the first ten, second ten and final ten trials. As can be seen in Fig. 8, the median accuracy scores jumped from 70% to 90% from the first to the second segments and remained relatively stable after. The difference in the first and second segments were statistically significant at the 95% level ($p = 0.033$), in favour of the hypothesis that human beings learnt continuously from feedback. We also noted that the human subjects repeated their grasps in order to make a decision (especially when they were unsure). As future work, we plan to investigate using STORK-TC's uncertainty estimates in a similar "active-touch" strategy.

VI. Conclusions and Future Work

In this work, we presented STORK-GP, a flexible spatio-temporal online algorithm that uses a novel recursive kernel that enables feature selection or weighting. Our experiments on benchmark dynamical systems demonstrated that STORK-GP produced accuracies superior to state-of-the-art methods.

As our second contribution, we have used STORK-GP as a "building block" in a generative classifier for tactile learning. We collected tactile data for nine everyday objects using the iCub humanoid and have made this dataset freely available online for future studies. In contrast to prior work, our STORK-TC classifier learns online from feedback. Our experiments showed it produced accuracies comparable to state-of-the-art optimised offline methods (decision tree and SVM), while being fast enough to be used and updated in real-time. We further compared our online method with human-subjects which yielded some interesting insights relating to active-learning and sensor fusion.

Looking towards the future, we envision several improvements that would increase STORK-GP's utility; we are working on online optimisation of kernel hyper-parameters and to further decrease its computational cost. It would also be interesting to investigate alternative hierarchical architectures (e.g., [26], [27]) for modelling more complex time-series. A further development on our classifier would be to incorporate actions, so that the iCub can re-grasp an object when uncertain about its class. Finally, we note that although we have focussed mainly on tactile classification, STORK-GP (and the generative classifier) can be applied in a variety of problems in robotics, ranging from learning-by-demonstration [28] to reinforcement learning.

Acknowledgement

The authors thank members of the Personal Robotics Laboratory at Imperial College London, particularly Yan Wu for his assistance in the iCub experiments. Harold Soh is supported by a Khazanah Global Scholarship.

References

[1] R. Neal, *Bayesian learning for neural networks*. Springer Verlag, 1996, vol. 118.

[2] G. Metta, G. Sandini, D. Vernon, L. Natale, and F. Nori, "The iCub humanoid robot: an open platform for research in embodied cognition," in *8th Workshop on Performance Metrics for Intelligent Systems*. New York, NY, USA: ACM, 2008, pp. 50–56.

[3] S. Chitta, M. Piccoli, and J. Sturm, "Tactile object class and internal state recognition for mobile manipulation," in *Robotics and Automation, IEEE International Conference on*, May 2010, pp. 2342 –2348.

[4] S. Chitta, J. Sturm, M. Piccoli, and W. Burgard, "Tactile sensing for mobile manipulation," *IEEE Trans. on Robotics*, vol. 27, no. 3, pp. 558 –568, June 2011.

[5] L. Csató, "Gaussian processes: iterative sparse approximations," Ph.D. dissertation, Aston University, 2002.

[6] L. Csató and M. Opper, "Sparse on-line gaussian processes," *Neural Computation*, vol. 14, no. 3, pp. 641–668, March 2002.

[7] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[8] K. M. A. Chai, C. K. I. Williams, S. Klanke, and S. Vijayakumar, "Multi-task gaussian process learning of robot inverse dynamics." in *NIPS*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds. Curran Associates, Inc., 2008, pp. 265–272.

[9] M. Schneider and W. Ertel, "Robot learning by demonstration with local gaussian process regression," in *IEEE/RSJ International Conference on Intelligent Robots and Systems,*, 2010, pp. 255 –260.

[10] D. Grollman and O. Jenkins, "Sparse incremental learning for interactive robot control policy estimation," in *IEEE International Conference on Robotics and Automation*, may 2008, pp. 3315 –3320.

[11] J. Quiñonero Candela and C. E. Rasmussen, "A unifying view of sparse approximate gaussian process regression," *J. Mach. Learn. Res.*, vol. 6, pp. 1939–1959, December 2005.

[12] H. Soh and Y. Demiris, "Iterative temporal learning and prediction with the sparse online echo state gaussian process," in *IEEE International Joint Conference on Neural Networks (IJCNN) - to appear*, 2012.

[13] M. Hermans and B. Schrauwen, "Recurrent kernel machines: Computing with infinite echo state networks," *Neural Computation*, vol. 24, no. 1, pp. 104–133, 2011.

[14] H. Jaeger, "The "echo state" approach to analysing and training recurrent neural networks," German National Research Center for Information Technology, Bremen, Tech. Rep. 148, 2001.

[15] H. Soh, "Online Temporal Learning (OTL) C++ Library." [Online]. Available: http://www.bitbucket.org/haroldsoh/otl

[16] S. Vijayakumar, A. D'Souza, and S. Schaal, "LWPR: Locally Weighted Projection Regression library."

[17] H. Soh, "Time series experimental setup." [Online]. Available: https://bitbucket.org/haroldsoh/oesgp_experiment_setup

[18] D. Verstraeten and M. Wardermann, "Reservoir computing toolbox." [Online]. Available: http://reslab.elis.ugent.be/rctoolbox

[19] E. Wen, "Time series data." [Online]. Available: http://web.cecs.pdx.edu/%7Eericwan/time-series-data.html

[20] S. Klanke and S. Vijayakumar, "LWPR supplementary documentation," Tech. Rep., 2008.

[21] H. Jaeger, "Adaptive nonlinear system identification with echo state networks," in *Advances in neural information processing system*, 2003, pp. 593–600.

[22] M. Schopfer, M. Pardowitz, and H. Ritter, "Using entropy for dimension reduction of tactile data," in *International Conference on Advanced Robotics*, June 2009, pp. 1 –6.

[23] M. Schopfer, H. Ritter, and G. Heidemann, "Acquisition and application of a tactile database," in *IEEE International Conference on Robotics and Automation*, april 2007, pp. 1517 –1522.

[24] A. Schneider, J. Sturm, C. Stachniss, M. Reisert, H. Burkhardt, and W. Burgard, "Object identification with tactile sensors using bag-of-features," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009, pp. 243 –248.

[25] H. Soh, "iCub Grasp Dataset." [Online]. Available: http://www.bitbucket.org/haroldsoh/icub_grasp_dataset

[26] M. Sarabia, R. Ros, and Y. Demiris, "Towards an open-source social middleware for humanoid robots," in *11th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, Oct. 2011, pp. 670 – 675.

[27] Y. Demiris and B. Khadhouri, "Hierarchical attentive multiple models for execution and recognition of actions," *Robotics and Autonomous Systems*, vol. 54, no. 5, pp. 361–369, 2006.

[28] Y. Wu and Y. Demiris, "Towards One Shot Learning by Imitation for Humanoid Robots," in *Proceedings of 2010 IEEE International Conference on Robotics and Automation. ICRA 2010.*, May 2010, pp. 2889–2894.