Real-Time Information Processing of Environmental Sensor Network Data using Bayesian Gaussian Processes[1]

M. A. Osborne and S. J. Roberts
Department of Engineering Science
University of Oxford
Oxford, OX1 3PJ, UK.
{mosb,sjrob}@robots.ox.ac.uk

and

A. Rogers and N. R. Jennings
School of Electronics and Computer Science
University of Southampton
Southampton, SO17 1BJ, UK.
{acr,nrj}@ecs.soton.ac.uk

In this paper, we consider the problem faced by a sensor network operator who must infer, in real-time, the value of some environmental parameter that is being monitored at discrete points in space and time by a sensor network. We describe a powerful and generic approach built upon an efficient multi-output Gaussian process that facilitates this information acquisition and processing. Our algorithm allows effective inference even with minimal domain knowledge, and we further introduce a formulation of Bayesian Monte Carlo to permit the principled management of the hyperparameters introduced by our flexible models. We demonstrate how our methods can be applied in cases where the data is delayed, intermittently missing, censored and/or correlated. We validate our approach using data collected from three networks of weather sensors and show that it yields better inference performance than both conventional independent Gaussian processes and the Kalman filter. Finally, we show that our formalism efficiently re-uses previous computations by following an online update procedure as new data sequentially arrives, and that this results in a four-fold increase in computational speed in the largest cases considered.

Categories and Subject Descriptors: C.2.4 [**Computer Communication Networks**]: Distributed Systems – Distributed Applications

General Terms: Algorithms, Design, Performance

Additional Key Words and Phrases: Learning of models from data, Gaussian processes, information processing, adaptive sampling

[1]A preliminary version of the work presented in this paper appeared as: Osborne, M. A., Rogers, A., Ramchurn, S., Roberts, S. J. and Jennings, N. R. (2008) Towards Real-Time Information Processing of Sensor Network Data using Computationally Efficient Multi-output Gaussian Processes. In: International Conference on Information Processing in Sensor Networks (IPSN 2008), April 2008, St. Louis, Missouri, USA. pp. 109-120.

1. INTRODUCTION

Sensor networks have recently generated a great deal of research interest within the computer and physical sciences, and their use for the scientific monitoring of remote and hostile environments is increasingly common-place. While early sensor networks were a simple evolution of existing automated data loggers that collected data for later off-line scientific analysis, more recent sensor networks now make their data available in real-time through the internet, and increasingly perform some form of real-time data processing or aggregation to provide useful summary information to users [Hart and Martinez 2006].

Providing real-time access to sensor data in this way presents many novel challenges; not least the need for self-describing data formats, and standard protocols such that the existence and capabilities of sensors can be advertised to potential users. However, more significantly for us, many of the information processing tasks that would previously have been performed off-line by the expert owner or single user of an environmental sensor network (such as detecting faulty sensors, performing 'data cleaning' to remove erroneous readings, fusing noisy measurements from several sensors, and deciding how frequently data should be collected), must now be performed autonomously in real-time. Furthermore, since such information processing is likely to be performed within centralised sensor repositories — of which Weather Underground (`http://www.wunderground.com`, pachube (http://www.pachube.com/) and Microsoft Research's SensorMap (`http://atom.research.microsoft.com/sensormap/`) are embryonic examples — and will be applied to open sensor networks where additional sensors may be deployed at any time, and existing sensors may be removed, repositioned or updated after deployment (such as the rooftop weather sensors within the Weather Underground), these information processing tasks may have to be performed with only limited knowledge of the precise location, reliability, and accuracy of each sensor.

Now, many of the information processing tasks described above have previously been tackled by applying principled Bayesian methodologies from the academic literature of geospatial statistics and machine learning: specifically, kriging [Cressie 1991] and Gaussian processes [Rasmussen and Williams 2006]. However, due to the computational complexity of these approaches, to date they have largely been used off-line in order to analyse and re-design existing sensor networks (e.g. to reduce maintenance costs by removing the least informative sensors from an existing sensor network [Fuentes et al. 2007], or to find the optimum placement of a small number of sensors, after a trial deployment of a larger number has collected data indicating their spatial correlation [Krause et al. 2006]). Alternatively, they have been applied to single sensors and have ignored the correlations that exist between different sensors within the network [Kho et al. 2009; Padhy et al. 2010]. Thus, there is a clear need for more computationally efficient algorithms, in order that this information processing can be performed at scale in real-time.

Against this background, this paper describes our work developing just such an algorithm. More specifically, we present a novel iterative formulation of a multi-output Gaussian process (GP) that uses a computationally efficient implementation of *Bayesian Monte Carlo* to marginalise hyperparameters, efficiently re-uses previous computations by following an online update procedure as new data sequentially

arrives, and uses a principled 'windowing' of data in order to maintain a reasonably sized data set. We use this GP to build a probabilistic model of the environmental variables being measured by sensors within the network, that is tolerant to data that may be missing, delayed, censored and/or correlated. This model allows us to then perform information processing tasks including: modelling the accuracy of the sensor readings, predicting the value of missing sensor readings, predicting how the monitored environmental variables will evolve in the near future, and performing active sampling by deciding when and from which sensor to acquire readings. We validate our approach using data collected from three networks of weather sensors and show that it yields better inference performance than both conventional independent single-output Gaussian processes, that model each sensor independently, and the Kalman filter. Finally, we analyse the computational speed of our algorithm. We show that our formalism efficiently re-uses previous computations by following an online update procedure as new data sequentially arrives, and this results in a four-fold increase in computational speed in the largest cases considered. Furthermore, it enables all of the information processing tasks described in this paper to be performed in real-time.

The remainder of this paper is organised as follows: Section 2 describes the information processing problem that we face. Section 3 presents our Gaussian process formulation, and Section 4 describes the two sensor networks used to validate it. In Section 5 we present experimental results using data from these networks, and in Section 6 we present results on the computational cost of our algorithm. Finally, related work is discussed in Section 7, and we conclude in Section 8.

2. THE INFORMATION PROCESSING PROBLEM

As discussed above, we require that our algorithm be able to autonomously perform data acquisition and information processing despite having only limited specific knowledge of each of the sensors in its local neighbourhood (e.g. their precise location, reliability, and accuracy). To this end, we require that it explicitly represents:

(1) The uncertainty in the estimated values of environmental variables being measured, noting that sensor readings will always incorporate some degree of measurement noise.
(2) The correlations or delays that exist between sensor readings; sensors that are close to one another, or in similar environments, will tend to make similar readings, while many physical processes involving moving fields (such as the movement of weather fronts) will induce delays between sensors.

We then require that it uses this representation in order to:

(1) Perform regression and prediction of environmental variables; that is, interpolate between sensor readings to predict variables at missing sensors (i.e. sensors that have failed or are unavailable through network outages), and perform short term prediction in order to support decision making.
(2) Perform efficient active sampling by selecting when to take a reading, and which sensor to read from, such that the minimum number of sensor readings are used to maintain the estimated uncertainty in environmental variables below a specified threshold (or similarly, to minimise uncertainty given a constrained

number of sensor readings). Such constraints may reflect the computational limitations of the processor on which the algorithm is running, or alternatively, where the algorithm is actually controlling the sensors within the network, it may reflect the constrained power consumption of the sensors themselves.

More specifically, the problem that we face can be cast as a multivariate regression and decision problem in which we have $l = 1, \ldots, L$ environmental variables $x_l \in \mathbb{R}$ of interest (such as air temperature, wind speed or direction specified at different sensor locations). We assume a set of N potentially noisy sensor readings, $\left\{\left[[l_1, t_1], z_1\right], \ldots, \left[[l_N, t_N], z_N\right]\right\}$, in which we, for example, observe the value z_1 for the ${l_1}^{\text{th}}$ variable at time t_1, whose true unknown value is y_1. Where convenient, we may group the inputs as $x = [l, t]$. Note that we do not require that all the variables are observed at the same time, nor do we impose any discretisation of our observations into regularly spaced time steps. We define our vector of observations as $\boldsymbol{z}_d \triangleq [z_1, \ldots, z_N]$ of variables labelled by $\boldsymbol{l}_d \triangleq [l_1, \ldots, l_N]$ at times $\boldsymbol{t}_d \triangleq [t_1, \ldots, t_N]$. Given this data, we are interested in inferring the vector of values $\boldsymbol{y}_\star$ for any other vector of variables labelled by $\boldsymbol{l}_\star$ at times $\boldsymbol{t}_\star$. We might, for example, wish to predict the values of variables at locations we do not currently have readings for (perhaps due to sensor failure), or to forecast future values for a variable at a particular sensor.

In the sections to follow, we'll describe an an algorithm built around Gaussian processes to address this problem. We exploit whatever prior expectations we may have for the variation of the environment variables over time and sensor, by building a model that can, for example, express that a particular variable may have some (unknown) periodicity over time and that readings from the various sensors may be correlated with one another. Such strong expectations, however, are not necessary; our algorithm is flexible enough to learn a broad range of models. This is achieved by using a means of approximate integration to marginalise over the class of models that are compatible with our observations. In essence, we sample a number of reasonable models and weight the predictions made under each potential model by its likelihood given the data. Those predictions are made in a computationally efficient, iterative fashion as we receive ever more data, and given our principled Bayesian framework, provide a measure of the uncertainty in our estimates. By managing these uncertainties, our algorithm is also able to select for itself the most informative observations, recognising that observations may be costly and must be selected intelligently.

3. GAUSSIAN PROCESSES

Multivariate regression problems of the form described above are increasingly being addressed using Gaussian processes (GPs). These represent a powerful way to perform Bayesian inference about functions; we consider our environmental variables as just such a function [Rasmussen and Williams 2006]. This function takes as inputs the variable label and time pair x and produces as output the variable's value y. In this work, we will assume that our inputs are always known (e.g. our data is time-stamped), and will incorporate them into our background knowledge, or context, I. A GP is then a generalised multivariate Gaussian prior distribution

over the (potentially infinite number of) outputs of this function:

$$p(\,\boldsymbol{y} \mid \boldsymbol{\mu},\, K,\, I\,) \triangleq \mathrm{N}(\boldsymbol{y}; \boldsymbol{\mu}, \mathbf{K}) \\ \triangleq \frac{1}{\sqrt{\det 2\pi\mathbf{K}}} \exp\left(-\frac{1}{2}\,(\boldsymbol{y}-\boldsymbol{\mu})^{\mathrm{T}}\,\mathbf{K}^{-1}\,(\boldsymbol{y}-\boldsymbol{\mu})\right).$$

It is specified by prior mean and covariance functions, which generate $\boldsymbol{\mu}$ and $\mathbf{K}$. The multivariate Gaussian distribution is qualified for this role due to the fact that both its marginal probabilities and conditional probabilities are themselves Gaussian. This allows us to produce analytic posterior distributions for outputs of interest, conditioned on whatever sensor readings have been observed. Furthermore, this posterior distribution will have both a predictive mean and a variance to explicitly represent our uncertainty.

While the fundamental theory of GPs is well established (see Rasmussen and Williams [2006] for example), there is much scope for the development of computationally efficient implementations especially for applications such as sensor networks which involve large numbers of sensors collecting large quantities of timestamped data. To this end, in this work we present a novel on-line formalism of a multi-dimensional GP that allows us to model the correlations between sensor readings, and to update this model on-line as new observations are sequentially available. In the next sections we describe the covariance functions that we use to represent correlations and delays between sensor readings, the *Bayesian Monte Carlo* method that we use to marginalise the hyperparameters of these covariance functions, and how we efficiently update the model as new data is received, by reusing the results of previous computations, and applying a principled 'windowing' of our data series.

3.1 Mean and Covariance Functions

The prior mean of a GP represents whatever we expect for our function before seeing any data. Throughout this paper, we will take this as a function constant in time for each sensor, such that $\mu([l,t]) = \mu_l$. The *covariance function* of a GP specifies the correlation between any pair of outputs. This can then be used to generate a covariance matrix over our set of observations and predictants. Fortunately, there exist a wide variety of functions that can serve in this purpose [Abrahamsen 1997; Stein 2005], which can then be combined and modified in a further multitude of ways. This gives us a great deal of flexibility in our modelling of functions, with covariance functions available to model periodicity, delay, noise and long-term drifts.

Examples of covariance functions are given by the Matérn class [Rasmussen and Williams 2006], given by

$$K_{\mathrm{Mtn}}(t,t';h,w,\nu) \triangleq h^2\,\frac{2^{1-\nu}}{\Gamma(\nu)}\left(\sqrt{2\nu}\,r\right)^{\nu}\,\mathfrak{K}_\nu\left(\sqrt{2\nu}\,r\right) \tag{1}$$

where $r = \left|\frac{t-t'}{w}\right|$, $\nu > 0$ is a smoothness hyperparameter (larger ν implies smoother functions) and $\mathfrak{K}_\nu$ is the modified Bessel function. Fortunately, (1) simplifies for half-integer ν, to give, for the example of $\nu = {}^5\!/_2$

$$K_{\mathrm{Mtn}}(t,t';h,\,w,\,\nu=5/2) \triangleq h^2\left(1+\sqrt{5}r+\frac{5r^2}{3}\right)\exp\left(-\sqrt{5}r\right), \tag{2}$$

and its modification for periodic signals

$$K_{\text{per-Mtn}}(t,t';h,\,w,\,\nu=5/2) \triangleq h^2\left(1+\sqrt{5}r_{\text{per}}+\frac{5r_{\text{per}}^2}{3}\right)\exp\left(-\sqrt{5}r_{\text{per}}\right), \quad (3)$$

where $r_{\text{per}} = \sin\pi\left|\frac{t-t'}{w}\right|$. This modification of r to accommodate periodicity is readily made for other covariance functions.

There are many ways to construct a covariance over a multi-dimensional input space, such as our two-dimensional input space $x=[l,t]$. We will typically exploit the fact that a product of covariances is a covariance in its own right, and write

$$K\big([l,t],[l',t']\big) = K_{\text{t}}(t,t')\,K_{\text{l}}(l,l')\,,$$

taking independent covariance terms over time and sensor label. If the number of sensors is not too large, we can arbitrarily represent the covariance matrix K_{l} over sensor labels using the spherical decomposition [Pinheiro and Bates 1996]. This allows us to arbitrarily represent any possible covariance over sensor labels. For a covariance matrix W, we write

$$W = S\,\operatorname{diag}(\tau)\,. \quad (4)$$

Here S is an upper triangular matrix, whose nth column contains the spherical coordinates in $\mathbb{R}^n$ of a point on the hypersphere $\mathbb{S}^{n-1}$, followed by the requisite number of zeros. As an example, S for a four dimensional space is

$$S = \begin{bmatrix} 1 & \cos\theta_1 & \cos\theta_2 & & \cos\theta_4 & \\ 0 & \sin\theta_1 & \sin\theta_2\,\cos\theta_3 & & \sin\theta_4\,\cos\theta_5 & \\ 0 & 0 & \sin\theta_2\,\sin\theta_3 & & \sin\theta_4\,\sin\theta_5 & \cos\theta_6 \\ 0 & 0 & 0 & & \sin\theta_4\,\sin\theta_5 & \sin\theta_6 \end{bmatrix}. \quad (5)$$

This form ensures that $S^{\text{T}}S$ has ones across its diagonal and hence all other entries may be thought of as akin to correlation coefficients, lying between -1 and 1. Meanwhile, τ is a vector of the scales for each dimension. If W is the covariance matrix parameterising the Mahalanobis distance, τ represents a vector of input scales for each input, and the off-diagonal elements of $S^{\text{T}}S$ express the correlation amongst inputs. If instead we have used W as a parameterisation of $K_{\text{l}}([1,\ldots,L],[1,\ldots,L])$, τ gives a vector of output scales corresponding to the different values $l=[1,\ldots,L]$. $S^{\text{T}}S$ gives the correlation coefficients for each pair of values l might take.

These intuitive connections provide the motivation for the use of this parameterisation. Finally, note that the total number of hyperparameters required by the spherical parameterisation is $\frac{1}{2}\,n\,(n+1)$ if the side length of W is n. Note that this is the same number as would be required to parameterise the non-zero elements of an upper triangular Cholesky factor $R=\operatorname{chol}W$ directly. The spherical parametrisation hence requires a large number of hyperparameters, but allows us to express any possible covariance matrix.

Finally, to represent measurement noise, we further extend the covariance function to

$$V([l,t],[l',t']) \triangleq K([l,t],[l',t']) + \sigma^2\,\delta([l,t]-[l',t'])\,,$$

where $\delta(-)$ is the Kronecker delta and σ^2 represents the variance of additive Gaussian noise.

The flexibility of our model comes at the cost of the introduction of a number of hyperparameters, which we collectively denote as ϕ. These include correlation hyperparameters, along with others such as the periods and amplitudes of each covariance term (i.e. w and h) and the noise deviation σ. The constant prior means $\mu_1, \ldots, \mu_L$ are also included as additional hyperparameters. Taking these hyperparameters as given and using the properties of the Gaussian distribution, we are able to write our predictive equations as

$$p(\, \boldsymbol{x}_\star \mid \boldsymbol{y}_d,\, \phi,\, I\,) = \mathrm{N}(\boldsymbol{x}_\star; \boldsymbol{m}_*, \mathbf{C}_*)\ , \tag{6}$$

where, collecting our inputs as $\boldsymbol{x}_\star \triangleq [\boldsymbol{l}_\star, \boldsymbol{t}_\star]$ and $\boldsymbol{x}_d \triangleq [\boldsymbol{l}_d, \boldsymbol{t}_d]$, we have:

$$\boldsymbol{m}_* = \boldsymbol{\mu}(\boldsymbol{x}_\star; \phi) + \mathbf{K}(\boldsymbol{x}_\star, \boldsymbol{x}_d; \phi)\mathbf{V}(\boldsymbol{x}_d, \boldsymbol{x}_d; \phi)^{-1}(\boldsymbol{y}_d - \boldsymbol{\mu}(\boldsymbol{x}_d; \phi)) \tag{7}$$

$$\mathbf{C}_* = \mathbf{K}(\boldsymbol{x}_\star, \boldsymbol{x}_\star; \phi) - \mathbf{K}(\boldsymbol{x}_\star, \boldsymbol{x}_d; \phi)\mathbf{V}(\boldsymbol{x}_d, \boldsymbol{x}_d; \phi)^{-1}\mathbf{K}(\boldsymbol{x}_d, \boldsymbol{x}_\star; \phi)\,. \tag{8}$$

For notational convenience, we will often drop the dependence upon ϕ from our covariance and mean functions.

3.2 Marginalisation

Of course, it is rare that we can be certain a priori about the values of our hyperparameters. One common approach to dealing with this uncertainty is to explore the possible space of hyperparameters (using a local search method such as gradient ascent) in order to select the set that exhibit the greatest log likelihood [Rasmussen and Williams 2006]. However, this approach fails to represent the uncertainty in the value of these hyperparameters, and thus, it fails to represent the true uncertainty in the final predictions of the environmental variables beings measured.

To correctly represent this uncertainty we must marginalise over ϕ, and thus, rather than equation (6), we must consider

$$p(\, \boldsymbol{y}_\star \mid \boldsymbol{z}_d, I\,) = \frac{\int \mathrm{d}\phi\, p(\, \boldsymbol{y}_\star \mid \boldsymbol{z}_d, \phi, I\,)\, p(\, \boldsymbol{z}_d \mid \phi, I\,)\, p(\, \phi \mid I\,)}{\int \mathrm{d}\phi\, p(\, \boldsymbol{z}_d \mid \phi, I\,)\, p(\, \phi \mid I\,)}\,. \tag{9}$$

Unfortunately, both our likelihood $p(\, \boldsymbol{z}_d \mid \phi, I\,)$ and predictions $p(\, \boldsymbol{y}_\star \mid \boldsymbol{z}_d, \phi, I\,)$ exhibit non-trivial dependence upon ϕ and so our integrals are non-analytic. As such, we resort to quadrature, which inevitably involves evaluating the two quantities

$$\begin{aligned} q(\phi) &\triangleq p(\, \boldsymbol{y}_\star \mid \boldsymbol{z}_d,\, \phi,\, I\,) \\ r(\phi) &\triangleq p(\, \boldsymbol{z}_d \mid \phi,\, I\,) \end{aligned} \tag{10}$$

at a set of sample points $\boldsymbol{\phi}_s = \{\phi_1, \ldots, \phi_\eta\}$, giving $\boldsymbol{q}_s \triangleq q(\boldsymbol{\phi}_s)$ and $\boldsymbol{r}_s \triangleq r(\boldsymbol{\phi}_s)$. Of course, this evaluation is a computationally expensive operation. Defining the vector of all possible function inputs as $\boldsymbol{\phi}$, we clearly can't afford to evaluate $\boldsymbol{q} \triangleq q(\boldsymbol{\phi})$ or $\boldsymbol{r} \triangleq r(\boldsymbol{\phi})$. Note that the more complex our model, and hence the greater the number of hyperparameters, the higher the dimension of the hyperparameter space we must sample in. As such, the complexity of models we can practically consider is limited by the curse of dimensionality. We can view our sparse sampling as introducing a form of uncertainty about the functions q and r, which we can again address using Bayesian probability theory.

To this end, we apply *Bayesian Monte Carlo*, and thus, assign further GP priors to q and r [Rasmussen and Ghahramani 2003]. This choice is motivated by the fact that variables over which we have a multivariate Gaussian distribution are joint Gaussian with any projections of those variables. As such, given that integration is a projection, we can use our computed samples $\boldsymbol{q}_s$ in order to perform Gaussian process regression about the value of integrals over $q(\phi)$, and similarly for r. Note that the quantity we wish to perform inference about,

$$\psi \triangleq p(\, \boldsymbol{y}_\star \mid \boldsymbol{q},\, \boldsymbol{r},\, \boldsymbol{z}_d, I\,) = \frac{\int \mathrm{d}\phi_\star\, q(\phi_\star)\, r(\phi_\star)\, p(\, \phi_\star \mid I\,)}{\int \mathrm{d}\phi_\star\, r(\phi_\star)\, p(\, \phi_\star \mid I\,)}\,, \tag{11}$$

possesses richer structure than that previously considered using Bayesian Monte Carlo techniques. In our case, $r(\phi)$ appears in both our numerator and denominator integrals, introducing correlations between the values we estimate for them. The correlation structure of this system is illustrated in figure 1.

In considering any problem of inference, we need to be clear about both what information we have and which uncertain variables we are interested in. In our case, both function values, $\boldsymbol{q}_s$ and $\boldsymbol{r}_s$, and their locations, $\boldsymbol{\phi}_s$, represent valuable pieces of knowledge[2]. As with our convention above, we will take knowledge of sample locations $\boldsymbol{\phi}_s$ to be implicit within I. We respectively define $\boldsymbol{m}^{(q)}$ and $\boldsymbol{m}^{(r)}$ as the means for $\boldsymbol{q}$ and $\boldsymbol{r}$ conditioned on $\boldsymbol{q}_s$ and $\boldsymbol{r}_s$ from (7), $\mathbf{C}^{(r)}$ and $\mathbf{C}^{(r)}$ the similarly conditional covariances from (8). The ultimate quantity of our interest is then

$$\begin{aligned}
&p(\, \boldsymbol{y}_\star \mid \boldsymbol{q}_s,\, \boldsymbol{r}_s,\, \boldsymbol{z}_d,\, I\,)\\
&= \iiint p(\, \boldsymbol{y}_\star \mid \boldsymbol{q},\, \boldsymbol{r},\, \boldsymbol{z}_d, I\,)\; p(\, \psi \mid \boldsymbol{q},\, \boldsymbol{r},\, I\,)\; p(\, \boldsymbol{q} \mid \boldsymbol{q}_s,\, I\,)\; p(\, \boldsymbol{r} \mid \boldsymbol{r}_s,\, I\,)\; \mathrm{d}\psi\, \mathrm{d}\boldsymbol{q}\, \mathrm{d}\boldsymbol{r}\\
&= \iiint \psi\, \delta\left(\psi - \frac{\int \mathrm{d}\phi_\star\, q_*\, r_*\, p(\, \phi_\star \mid I\,)}{\int \mathrm{d}\phi_\star\, r_*\, p(\, \phi_\star \mid I\,)}\right) \mathrm{N}\Big(\boldsymbol{q}; \boldsymbol{m}^{(q)}, \mathbf{C}^{(q)}\Big)\; \mathrm{N}\Big(\boldsymbol{r}; \boldsymbol{m}^{(r)}, \mathbf{C}^{(r)}\Big)\; \mathrm{d}\psi\, \mathrm{d}\boldsymbol{q}\, \mathrm{d}\boldsymbol{r}\\
&= \int \frac{\int \mathrm{d}\phi_\star\, m_*^{(q)}\, r_*\, p(\, \phi_\star \mid I\,)}{\int \mathrm{d}\phi_\star\, r_*\, p(\, \phi_\star \mid I\,)}\, \mathrm{N}\Big(\boldsymbol{r}; \boldsymbol{m}^{(r)}, \mathbf{C}^{(r)}\Big)\; \mathrm{d}\boldsymbol{r}\,.
\end{aligned} \tag{12}$$

Here, unfortunately, our integration over r becomes nonanalytic. However, we can employ a maximum *a posteriori* approximation by assuming

$$\mathrm{N}\Big(\boldsymbol{r}; \boldsymbol{m}^{(r)}, \mathbf{C}^{(r)}\Big) \simeq \delta\Big(\boldsymbol{r} - \boldsymbol{m}^{(r)}\Big)\;.$$

Essentially, we assume that our uncertainty in the likelihood function is small and trust that our more correct treatment of the uncertainty in the prediction function q is sufficient. Given that we are likely to be uncertain about $r(\phi)$ at exactly the same ϕ we are uncertain about $q(\phi)$, those points far removed from our samples $\boldsymbol{\phi}_s$, this approximation is not unreasonable. Before we can state our consequent results, to each of our hyperparameters we assign a Gaussian prior distribution (or if our hyperparameter is restricted to the positive reals, we instead assign a Gaussian distribution to its log) given by

$$p(\, \phi \mid I\,) \triangleq \mathrm{N}\big(\phi; \boldsymbol{\nu}, \lambda^{\mathrm{T}}\lambda\big)\;. \tag{13}$$

[2]As discussed by O'Hagan [1987], traditional, frequentist Monte Carlo effectively ignores the information content of $\boldsymbol{\phi}_s$, leading to several unsatisfactory features.

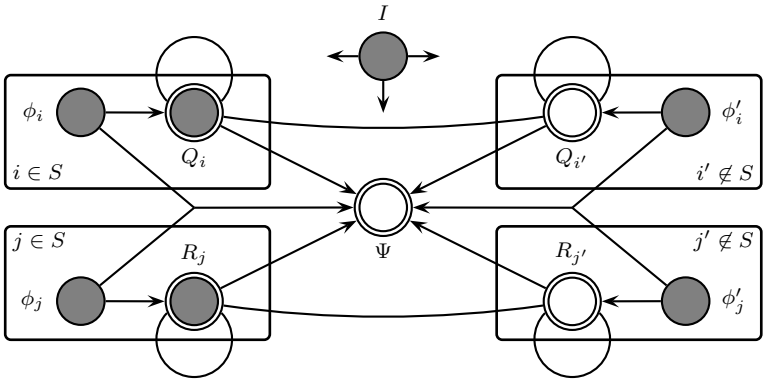

Fig. 1. Bayesian network for marginalising hyperparameters using Bayesian Monte Carlo. Shaded nodes are known and double-circled nodes are deterministic given all their parents. All Q nodes are correlated with one another, as are all R nodes, and the context I is correlated with all nodes.

Note that the intuitive spherical parameterisation (5) assists with the elicitation of priors over its hyperparameters. We then assign a *squared exponential* covariance function for the GP over both q and r given by

$$K(\phi, \phi') \triangleq \mathrm{N}\big(\phi; \phi', \mathbf{w}^{\mathrm{T}}\mathbf{w}\big) \ . \tag{14}$$

Finally, using the further definition for $i, j \in \mathcal{I}_s$, that

$$\mathfrak{N}_s(i,j) \triangleq \mathrm{N}\left(\begin{bmatrix} \boldsymbol{\phi}_i \\ \boldsymbol{\phi}_j \end{bmatrix}; \begin{bmatrix} \boldsymbol{\nu} \\ \boldsymbol{\nu} \end{bmatrix}, \begin{bmatrix} \lambda^{\mathrm{T}}\lambda + \mathbf{w}^{\mathrm{T}}\mathbf{w} & \lambda^{\mathrm{T}}\lambda \\ \lambda^{\mathrm{T}}\lambda & \lambda^{\mathrm{T}}\lambda + \mathbf{w}^{\mathrm{T}}\mathbf{w} \end{bmatrix} \right) , \tag{15}$$

our approximation gives us

$$p(\, \boldsymbol{y}_\star \mid \boldsymbol{y}_d, I\,) \simeq \boldsymbol{q}_s^{\mathrm{T}}\, \boldsymbol{\rho}\,, \tag{16}$$

where the weights of this linear combination are

$$\boldsymbol{\rho} \triangleq \frac{\mathbf{K}(\boldsymbol{\phi}_s, \boldsymbol{\phi}_s)^{-1}\, \mathfrak{N}_s\, \mathbf{K}(\boldsymbol{\phi}_s, \boldsymbol{\phi}_s)^{-1}\, \boldsymbol{r}_s}{\mathbf{1}_{s,1}^{\mathrm{T}}\, \mathbf{K}(\boldsymbol{\phi}_s, \boldsymbol{\phi}_s)^{-1}\, \mathfrak{N}_s\, \mathbf{K}(\boldsymbol{\phi}_s, \boldsymbol{\phi}_s)^{-1}\, \boldsymbol{r}_s}\,, \tag{17}$$

and $\mathbf{1}_{s,1}$ is a vector containing only ones of dimensions equal to $\boldsymbol{q}_s$. With a GP on $p(\, \boldsymbol{y}_\star \mid \phi, I\,)$, each $q_i = p(\, \boldsymbol{y}_\star \mid \boldsymbol{z}_d, \phi_i, I\,)$ will be a slightly different Gaussian. Hence we effectively approximate $p(\, \boldsymbol{y}_\star \mid \boldsymbol{z}_d, I\,)$ as a Gaussian (process) mixture; Bayesian Monte Carlo returns a weighted sum of our predictions evaluated at a sample set of hyperparameters. The assignment of these weights is informed by the best use of all pertinent information. As such, it avoids the risk of overfitting that occurs when applying a less principled technique such as maximum likelihood [MacKay 2002].

3.3 Censored Observations

In the work above, we have assumed that observations of our variables of interest were corrupted by simple Gaussian noise. However, in many contexts, we instead observe *censored* observations. That is, we might observe that a variable was above or below certain thresholds, but no more. Examples are rich within the weather

sensor networks considered. Float sensors are prone to becoming lodged on sensor posts, reporting only that the water level is below that at which it is stuck. Conversely, wind sensors may reach their maximum reading in very strong winds, reporting that the wind is greater than this limit, but providing no more information. Furthermore, readings are often rounded to the nearest integer, meaning that if we observe a reading of x, we can say only that the true value was between $x-0.5$ and $x+0.5$. We can readily extend our inference framework to allow for such a noise model.

More precisely, we assume that we actually observe bounds $\boldsymbol{b}_c$ that constrain Gaussian-noise corrupted versions $\boldsymbol{z}_c$ of the underlying variables of interest $\boldsymbol{y}_c$ at $\boldsymbol{x}_c$. This framework allows for imprecise censored observations. Note that the noise variance for censored observations may differ from the noise variance associated with other observations. Conditioned on a combination of censored and un-censored observations, the distribution for our variables of interest is

$$p(\,\boldsymbol{y}_\star \mid \boldsymbol{z}_d, \boldsymbol{b}_c, I\,) = \frac{\int \mathrm{d}\phi \int_{\boldsymbol{b}_c} \mathrm{d}\boldsymbol{z}_c\, p(\,\boldsymbol{y}_\star \mid \boldsymbol{z}_d, \boldsymbol{z}_c, \phi, I\,)\, p(\,\boldsymbol{z}_c \mid \boldsymbol{z}_d, \phi, I\,)\, p(\,\boldsymbol{z}_d \mid \phi, I\,)\, p(\,\phi \mid I\,)}{\int \mathrm{d}\phi \int_{\boldsymbol{b}_c} \mathrm{d}\boldsymbol{z}_c\, p(\,\boldsymbol{z}_c \mid \boldsymbol{z}_d, \phi, I\,)\, p(\,\boldsymbol{z}_d \mid \phi, I\,)\, p(\,\phi \mid I\,)}\,.$$

While we cannot determine this full, non-Gaussian distribution easily, we can determine the analytic forms for its mean and covariance. We use the abbreviations $\boldsymbol{m}_{c|d,\phi} \triangleq \boldsymbol{m}(\boldsymbol{z}_c|\boldsymbol{z}_d, \phi, I)$ and $\mathbf{C}_{c|d,\phi} \triangleq \mathbf{C}(\boldsymbol{z}_c|\boldsymbol{z}_d, \phi, I)$. To reflect the influence of our censored observations, our likelihoods become

$$p(\,\boldsymbol{z}_d,\, \boldsymbol{b}_c \mid \phi,\, I\,) = \mathrm{N}(\boldsymbol{z}_d; \boldsymbol{m}(\boldsymbol{z}_d|\phi, I)\,, \mathbf{C}(\boldsymbol{z}_d|\phi, I)) \int_{\boldsymbol{b}_c} \mathrm{d}\boldsymbol{z}_c\, \mathrm{N}\big(\boldsymbol{z}_c; \boldsymbol{m}_{c|d,\phi}, \mathbf{C}_{c|d,\phi}\big)\,,$$

giving the new weights over hyperparameter samples $\boldsymbol{\rho}^{(cd)}$. We can then write our predictive mean as

$$\boldsymbol{m}(\boldsymbol{y}_\star \,|\, \boldsymbol{z}_d, \boldsymbol{b}_c, I) = \sum_{i\in s} \rho_i^{(cd)} \Bigg(\boldsymbol{\mu}(\boldsymbol{x}_\star; \phi_i) + \mathbf{K}(\boldsymbol{x}_\star, [\boldsymbol{x}_c, \boldsymbol{x}_d]; \phi_i) \mathbf{V}([\boldsymbol{x}_c, \boldsymbol{x}_d], [\boldsymbol{x}_c, \boldsymbol{x}_d]; \phi_i)^{-1} \begin{bmatrix} \frac{\int_{\boldsymbol{b}_c} \mathrm{d}\boldsymbol{z}_c\, \boldsymbol{z}_c\, \mathrm{N}\left(\boldsymbol{z}_c; \boldsymbol{m}_{c|d,\phi_i}, \mathbf{C}_{c|d,\phi_i}\right)}{\int_{\boldsymbol{b}_c} \mathrm{d}\boldsymbol{z}_c\, \mathrm{N}\left(\boldsymbol{z}_c; \boldsymbol{m}_{c|d,\phi_i}, \mathbf{C}_{c|d,\phi_i}\right)} - \boldsymbol{\mu}(\boldsymbol{y}_c; \phi_i) \\ \boldsymbol{z}_d - \boldsymbol{\mu}(\boldsymbol{x}_d; \phi_i) \end{bmatrix} \Bigg)\,,$$

noting that a censored observation is intuitively treated as an uncensored observation equal to the conditional mean of the GP over the bounded region. We have also the predictive covariance

$$\mathbf{C}(\boldsymbol{y}_{st} \,|\, \boldsymbol{z}_d, \boldsymbol{b}_c, I) = \sum_{i\in s} \rho_i^{(cd)} \Big(\mathbf{K}(\boldsymbol{x}_\star, \boldsymbol{x}_\star; \phi_i) - \mathbf{K}(\boldsymbol{x}_\star, [\boldsymbol{x}_c, \boldsymbol{x}_d]; \phi_i) \mathbf{V}([\boldsymbol{x}_c, \boldsymbol{x}_d], [\boldsymbol{x}_c, \boldsymbol{x}_d]; \phi_i)^{-1} \mathbf{K}([\boldsymbol{x}_c, \boldsymbol{x}_d], \boldsymbol{y}_\star; \phi_i) \Big)\,.$$

We now have the problem of determining the integrals

$$\int_{\boldsymbol{b}_c} \mathrm{d}\boldsymbol{z}_c\, \mathrm{N}\big(\boldsymbol{z}_c; \boldsymbol{m}_{c|d,\phi}, \mathbf{C}_{c|d,\phi}\big) \qquad \text{and} \qquad \int_{\boldsymbol{b}_c} \mathrm{d}\boldsymbol{z}_c\, \boldsymbol{z}_c\, \mathrm{N}\big(\boldsymbol{z}_c; \boldsymbol{m}_{c|d,\phi}, \mathbf{C}_{c|d,\phi}\big)\,,$$

which are non-analytic. With only a single censored observation, we can use standard functions for such integrals. Assuming that our censored observations are independent of one another (conditioned on the other observations $\boldsymbol{z}_d$) would also allow us the use of such results, but is usually an unreasonable approximation. Fortunately, there also exists an efficient Monte Carlo algorithm [Genz 1992] for the purpose of resolving such integrals. It is this algorithm that we will choose for the appropriate empirical tests to follow.

3.4 Efficient Implementation

Now the implementation of equations (7) and (8) involves the inverse of a matrix, and the most stable way to implement this is through the use of the Cholesky decomposition, $\mathbf{R}(\boldsymbol{x}_d, \boldsymbol{x}_d)$, of $\mathbf{V}(\boldsymbol{x}_d, \boldsymbol{x}_d)$, rather than inverting the matrix directly. Performing this Cholesky decomposition represents the most computationally expensive operation we must perform; its cost scaling as $O(N^3)$ in the number of data points N. However, as discussed earlier, we do not intend to use our GP with a fixed set of data, but rather, within an on-line algorithm that receives new observations over time. As such, we must be able to iteratively update our predictions in as little time as possible. Fortunately, we can do so by exploiting the special structure of our problem. When we receive new data, our $\mathbf{V}$ matrix is changed only in the addition of a few new rows and columns. Hence most of the work that went into computing its Cholesky decomposition at the last iteration can be recycled to produce the new Cholesky decomposition (see Appendix A.1 for details of this operation). Another problematic calculation required by (7) and (8) is the computation of the data-dependent term $\mathbf{R}(\boldsymbol{x}_d, \boldsymbol{x}_d)^{-1}(\boldsymbol{y}_d - \boldsymbol{\mu}(\boldsymbol{x}_d))$, in which $\boldsymbol{y}_d - \boldsymbol{\mu}(\boldsymbol{x}_d)$ is also only changing due to the addition of new rows. As such, efficient updating rules are also available for this term (see Appendix A.2). Finally, we are also able to perform an efficient update to the likelihood[3] of our hyperparameters, as per Appendix A.3. As per (17), the updated sample likelihoods $\boldsymbol{r}_s$ then give rise to updated weights ρ over our hyperparameter samples. With these efficient rules, we are able to reduce the overall cost of an update from $O(N^3)$ to $O(N^2)$.

However, we can further increase the efficiency of our updates by making a judicious assumption. In particular, experience shows that our GP requires only a very small number of recent observations in order to produce good estimates. Indeed, most covariance functions have very light tails such that only points within a few multiples of the time scale are at all relevant to the point of interest. Hence we seek sensible ways of discarding information once it has been rendered 'stale', to reduce both memory usage and computational requirements.

One pre-eminently reasonable measure of the value of data is the uncertainty we still possess after learning it. In particular, we are interested in how uncertain we are about $\boldsymbol{x}_\star$; as given by the covariance of our Gaussian mixture equation (16). Our approach is thus to drop our oldest data points (those which our covariances typically deem least relevant to the current predictant) until this uncertainty exceeds some predetermined threshold.

Just as we were able to efficiently update our Cholesky factor upon the receipt

[3]Note that likelihoods are commonly smaller than machine precision and so for numerical reasons we prefer to instead work with log-likelihoods, exponentiating as required.

of new data, so we can downdate to remove data (see Appendix A.4 for the details of this operation). Unfortunately, there is no such efficient downdate for the data-dependent term $\mathbf{R}(\boldsymbol{x}_d, \boldsymbol{x}_d)^{-1}(\boldsymbol{y}_d - \boldsymbol{\mu}(\boldsymbol{x}_d))$. Naturally there is no computational benefit here to be gained in downdating our hyperparameter likelihoods, whose dimension remains constant regardless of the size of the dataset. Hence our likelihoods are never downdated, and we retain all knowledge of otherwise dropped data that is pertinent to our marginalisation of hyperparameters.

Our proposal allows us to rapidly remove unwanted data, compute our uncertainty about $\boldsymbol{y}_\star$, and then repeat as required; the GP will retain only as much data as necessary to achieve a pre-specified degree of accuracy. This allows a principled way of 'windowing' our data series.

Finally, we turn to the implementation of our marginalisation procedure. Essentially, our approach is to maintain an ensemble of GPs, one for each hyperparameter sample, each of which we update and downdate according to the proposals above. Note that the computations for each hyperparameter sample can be treated independently, lending our algorithm to parallelisation.

The predictions of each sample are then weighted and combined according to equation (16). Note that the only computations whose computational cost grows at greater than a quadratic rate in the number of samples, η, are the Cholesky decomposition and multiplication of covariance matrices in equation (17), and these scale rather poorly as $O(\eta^3)$. To address this problem, we take our Gaussian priors for each different hyperparameter $\phi_{(e)} \in \phi$ as independent. We further take a covariance structure given by the product of terms over each hyperparameter, the common *product correlation rule* (e.g. Sasena [2002])

$$K(\phi, \phi') = \prod_e K_e\Big(\phi_{(e)}, \phi'_{(e)}\Big). \tag{18}$$

If we additionally consider a simple grid of samples, such that $\boldsymbol{\phi}_s$ is the tensor product of a set of samples $\boldsymbol{\phi}_{(e),s}$ over each hyperparameter, then the problematic term in equation (16) reduces to the Kronecker product of the equivalent term over each individual hyperparameter:

$$\begin{aligned}\mathbf{K}(\boldsymbol{\phi}_s, \boldsymbol{\phi}_s)^{-1}\, \mathfrak{N}_s\, \mathbf{K}(\boldsymbol{\phi}_s, \boldsymbol{\phi}_s)^{-1} \\ = \mathbf{K}(\boldsymbol{\phi}_{(1),s}, \boldsymbol{\phi}_{(1),s})^{-1} \mathfrak{N}_s(\boldsymbol{\phi}_{(1),s}, \boldsymbol{\phi}_{(1),s}) \mathbf{K}(\boldsymbol{\phi}_{(1),s}, \boldsymbol{\phi}_{(1),s})^{-1} \\ \otimes \mathbf{K}(\boldsymbol{\phi}_{(2),s}, \boldsymbol{\phi}_{(2),s})^{-1} \mathfrak{N}_s(\boldsymbol{\phi}_{(2),s}, \boldsymbol{\phi}_{(2),s}) \mathbf{K}(\boldsymbol{\phi}_{(2),s}, \boldsymbol{\phi}_{(2),s})^{-1} \\ \otimes \dots\end{aligned} \tag{19}$$

This means that we only have to perform the expensive Cholesky factorisation and multiplication with matrices whose size equals the number of samples for each hyperparameter, rather than on a matrix of size equal to the total number of hyperparameter samples. This hence represents an effective way to avoid the 'curse of dimensionality'.

Note that we can also leverage some of the benefits of maximum likelihood to our own advantage. In particular, given an initial grid of samples, we can perform regular updates to their positions by performing single-stage gradient ascent. That is, the value of $\phi_{(e),i}$, the ith sample in the eth hyperparameter, is adjusted according to the average derivative with respect to $\phi_{(e),i}$ of the likelihood of all

samples that share $\phi_{(e),i}$. Shifting a hyperparameter sample does, however, require the fresh Cholesky decomposition of the covariance matrix associated with that sample, along with the re-computation of the data-dependent term, likelihood and the weights term (19). As such, we perform such sample shifting relatively rarely to avoid too excessive a computational burden.

In general, limiting the number of hyperparameter samples is of critical importance to achieving practical computation. As such, we should exploit any and all prior information that we possess about the system to limit the volume of hyperparameter space that our GP is required to explore online. For example, an informative prior expressing that a tidal period is likely to be around half a day will greatly reduce the number of samples required for this hyperparameter. Similarly, an offline analysis of any available training data will return sharply peaked posteriors over our hyperparameters that will further restrict the required volume to be searched over on-line. For example, we represent the tidal period hyperparameter with only a single sample on-line, so certain does training data make us of its value. Finally, a simpler and less flexible covariance model, with fewer hyperparameters, could be chosen if computational limitations become particularly severe.

Applied together, these features provide us with an efficient on-line algorithm that can be applied in real-time as data is sequentially collected from the sensor network. Pseudo-code for our tracking algorithm is displayed in algorithm 1.

3.5 Active Data Selection

Finally, in addition to the regression and prediction problem described in Section 2, we are able to use the same algorithm to perform active data selection. This is a decision problem concerning which observations should be taken. In this, we once again take a utility that is a function of the uncertainty in our predictions. We specify a utility of negative infinity if our uncertainty about any variable is greater than a pre-specified threshold, and a fixed negative utility is assigned as the cost of an observation (in general, this cost could be different for different sensors). Note that the uncertainty increases monotonically in the absence of new data, and shrinks in the presence of an observation. Hence our algorithm is simply induced to make a reading whenever the uncertainty grows beyond a pre-specified threshold.

Our algorithm can also decide which observation to make at this time, by determining which sensor will allow it the longest period of grace until it would be forced to observe again. This clearly minimises the number of costly observations. Note that the uncertainty of a GP, as given by (8), is actually dependent only on the location of the observation, not its actual value. Hence the uncertainty we imagine remaining after taking an observation from a sensor can be quickly determined without having to speculate about what data we might possibly collect. However, this is true only so long as we do not consider the impact of new observations on our hyperparameter sample weights (17). Our approach is to take the model, in particular the weights over samples, as fixed, and investigate only how different schedules of observations affect our predictions within it. With this proviso, we are guaranteed to maintain our uncertainty below a specified threshold, while taking as few observations as possible.

Algorithm 1 GP-TRACK($\boldsymbol{x}_{1:T}, \boldsymbol{z}_{1:T}, \delta$, MEAN$(\cdot;\cdot)$, COV$(\cdot,\cdot;\cdot), \boldsymbol{\phi}_s$)

// Returns the sequential lookahead predictions made with our GP algorithm.
// $\{(x_t, z_t)\}_{t=1}^{T}$: All data.
// δ: Lookahead.
// MEAN(x, ϕ): Mean function (μ), taking as arguments an input x and
// hyperparameter vector ϕ.
// COV$(x, x'; \phi)$: Covariance function (K), taking as arguments two inputs
// x and x' and hyperparameter vector ϕ.
// $\boldsymbol{\phi}_s \triangleq \{\phi_j\}_{i=1}^{\eta}$: Hyperparameter samples.

$\boldsymbol{x}_d \leftarrow \{\}$
$\boldsymbol{z}_d \leftarrow \{\}$
for $i = 1$ **to** η **do**
  $GP[i] \leftarrow$ INITIALISE-GP(MEAN, COV, ϕ_i) // Supply $GP[i]$ with the mean function MEAN(x, ϕ_i) and covariance function COV$(x, x'; \phi_i)$.
end for
weights-term $\leftarrow$ CALCULATE-WEIGHTS-TERM($\boldsymbol{\phi}_s$) // Calculate the weights term from (19), $\mathbf{K}(\boldsymbol{\phi}_s, \boldsymbol{\phi}_s)^{-1}\, \mathfrak{N}_s\, \mathbf{K}(\boldsymbol{\phi}_s, \boldsymbol{\phi}_s)^{-1}$.
for $t = 1$ **to** $T - \delta$ **do**
  ($\boldsymbol{x}_d, \boldsymbol{z}_d$, *drop-inds, add-ind*) $\leftarrow$ INCREMENT-DATA($(\boldsymbol{x}_d, \boldsymbol{z}_d), (x_t, z_t), GP$) // Add new datum (index *add-ind*), and drop old data (indices *drop-inds*) as necessary (as per Section 3.4).
  for $i = 1$ **to** η **do**
    $GP[i] \leftarrow$ DOWNDATE-GP($GP[i]$,*drop-inds*) // Downdates a GP as per Section 3.4, revising covariance matrix and data-dependent term to allow for dropped data.
    $GP[i] \leftarrow$ UPDATE-GP($GP[i], (\boldsymbol{x}_d, \boldsymbol{z}_d)$,*add-ind*) // Updates a GP as per Section 3.4, revising covariance matrix, data-dependent term and likelihoods to allow for added data.
    $(m_{t+\delta,i}, C_{t+\delta,i}) \leftarrow$ MAKE-PREDICTIONS($x_{t+\delta}, GP[i]$) // Compute the posterior mean and variances for predictant $\boldsymbol{y}_\star = y_{t+\delta}$ using (7) and (8).
  end for
  $\boldsymbol{\rho} \leftarrow$ DETERMINE-WEIGHTS(*weights-term*, $GP[1:\eta]$) // Using (17).
  $(m'_{t+\delta}, C'_{t+\delta}) \leftarrow$ COMBINE-PREDICTIONS($\{\rho_i, m_{t+\delta,i}, C_{t+\delta,i}\}_{i=1}^{\eta}$) // Gaussians are combined in a weighted mixture, using (16).
  Return: $(m'_{t+\delta}, C'_{t+\delta})$
  if IS-TIME-TO-MOVE(t) // Infrequently true **then**
    ($\boldsymbol{\phi}_s$, *move-inds*) $\leftarrow$ MANAGE-HYPERSAMPLES($GP[1:\eta], \boldsymbol{\phi}_s, t$) // Move hyperparameter samples *move-inds*, as per Section 3.4.
    for $i \in$ *move-inds* **do**
      $GP[i] \leftarrow$ OVERWRITE-GP($GP[i], (\boldsymbol{x}_d, \boldsymbol{z}_d), \phi_i$) // Recompute all quantities given the new hyperperparameter sample.
    end for
    weights-term $\leftarrow$ CALCULATE-WEIGHTS-TERM($\boldsymbol{\phi}_s$)
  end if
end for

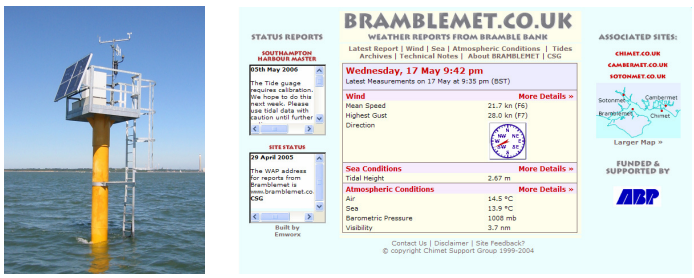

Fig. 2. The Bramble Bank weather station and web site.

4. TRIAL IMPLEMENTATION

In order to empirically evaluate the information processing algorithm described in the previous section, we have used data from three networks of weather sensors of different sizes (specifically, 4, 16 and 670 sensors) that provide a range of different sensor measurements. These networks are the Bramblemet sensor network, the Wannengrat Alpine Observatory, and the UK Met Office MIDAS land surface weather stations. The use of these weather sensor networks is attractive since they exhibit challenging correlations and delays whose physical processes are well understood. However, we stress that the approach that we describe in this paper is general, and can equally well be applied to any time series that exhibits complex correlations and delays.

4.1 Bramblemet Weather Sensor Network

The Bramblemet network consists of four sensors (named Bramblemet, Sotonmet, Cambermet and Chimet), each of which measures a range of environmental variables (including wind speed and direction, air temperature, sea temperature, and tide height) and makes up-to-date sensor measurements available through separate web pages (see figure 2 and `http://www.bramblemet.co.uk/`). To facilitate the autonomous collection of sensor data by our information processing algorithm, we have worked with the sensor network operators (Associated British Ports) and supplemented each sensor web page with machine readable RDF data (see figure 3 for an example of this format – current sensor data in this format is available at `http://www.bramblemet.co.uk/bra.rdf`). This format is attractive as it represents a fundamental element of the semantic web, and there exist a number of software tools to parse, store and query it. More importantly, it allows the sensor data to be precisely defined through standard ontologies [Lassila and Swick 1999]. For example, linking the predicate *geo:lat* to the ontology available at `http://www.w3.org/2003/01/geo/wgs84_pos#` precisely defines the value *"50.79472"* as representing a latitude in the WGS84 geodetic reference datum. While ontologies for sensor data have yet to be standardised, a number of candidates exist (see [Barnaghi et al. 2009] for an example based upon Sensor Web Enablement (SWE) and SensorML data component models).

In order to visualise the sensor data and the information processing algorithms in operation, we have implemented a server-based application that collects the

```
<sit:Location rdf:about="&sit;bramblemet"
  rdfs:label="Bramble Bank"
  geo:lat="50.79472"
  geo:lng="-1.2875"
  sit:altitude="1">
</sit:Location>

<sit:Sensor rdf:about="&sit;bramblemet/windspeed"
  rdfs:label="Wind speed">
  <sit:sensorType rdf:resource="&sit;windspeed"/>
  <sit:location rdf:resource="&sit;bramblemet"/>
</sit:Sensor>

<sit:SensorType rdf:about="&sit;windspeed"
  rdfs:label="Wind speed">
</sit:SensorType>

<sit:Unit rdf:about="&sit;knots"
  rdfs:label="Knots"
  sit:unitAbbr="kn">
</sit:Unit>

<sit:Reading
  rdf:about="&sit;bramblemet/windspeed/reading/1234"
  rdfs:value="9.3"
  sit:datetime="2007-10-25T21:55:00">
  <sit:sensor rdf:resource="&sit;bramblemet/windspeed"/>
  <sit:unit rdf:resource="&sit;knots"/>
</sit:Reading>
```

Fig. 3. Example RDF data from the Bramblemet sensor.

RDF data from the sensors, parses and stores it using Jena (see `http://jena.sourceforge.net/`), and displays sensor data in tabular and graphical forms using a Google Maps interface (see figure 4). A live version of this system is available at `http://www.aladdinproject.org/situation/`.

The use of the Bramblemet sensor network is particularly attractive since the network is subject to real network outages that generate instances of missing sensor readings on which we can evaluate our information processing algorithms. Furthermore, by working with the sensor network operators we have been able to recover the missing data from these periods from the sensors' local caches, and thus, perform quantitative comparisons between our information processing algorithm's predictions and the actual data. Furthermore, within the Bramblemet network sensors, the sensor measurements are rounded prior to their transmission over the wireless link. This results in censored observations, and thus, again by accessing the raw data from the sensors' local caches, we are also able to compare the ability of our information processing algorithms to handle these censored observations.

However, the Bramblemet sensor network only consists of four sensors, and thus, in order to compare the performance of our approach on larger networks, we turn to the Wannengrat Alpine observatory and the UK Met Office MIDAS land surface

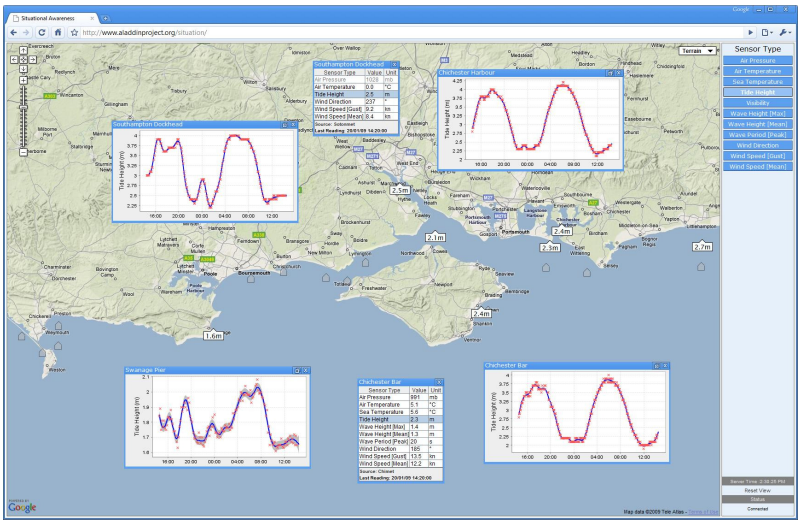

Fig. 4. Server-based implementation of our information processing algorithm accessed through a web based Google Maps interface (available at `http://www.aladdinproject.org/situation/`).

stations.

4.2 Wannengrat Alpine Observatory

The Wannengrat site consists of over twenty SensorScope stations to measure local weather conditions[4]. This wireless sensor network is deployed with the aim of understanding the complex meterological processes occurring on the snowpack. For this network, we simply use the data from online data repository, and have had no direct interaction with the sensor network operators.

4.3 MIDAS Land Surface Weather Stations

Finally, we use data from the UK Met Office MIDAS land surface stations data [British Atmospheric Data Centre 2009], focusing on the network's readings of maximum daily temperature at 670 sensors distributed across the UK. Our experiments on the data collected from these several hundred sensors serve as a demonstration of the ability of our algorithm to cope with larger sensor networks.

5. EMPIRICAL EVALUATION

In this section we empirically evaluate our information processing algorithm on real weather data collected from the sensor networks described above. We will compare the use of our GP formalism built around a multiple-sensor covariance against a number of benchmarks:

Conventional independent GP in which each environmental variable is modelled separately (i.e. correlations between these parameters are ignored).

[4]see `http://www.swiss-experiment.ch/index.php/Wannengrat:Sensorscopedeployment`.

Kalman filter in which a maximum a posteriori (MAP) recursion is used to govern the adaptions of the hyper-parameters (namely the state and observation noise processes) by sequentially using the maximum-likelihood formulation first proposed by Jazwinski [1970]. One step-ahead forecasts are successively iterated through the algorithm to provide multi-step forecasts of arbitrary length. Missing values, if they occur, are accommodated for using the scheme advocated by Shumway and Stoffer [2000]. The algorithm operates, for multiple time-series forecasting, using a p-th order lookback into the data, including cross-coupling between time-series, making it a recursive multivariate autoregressive, MAR(p), process.

Naïve algorithm in which the prediction of the variable at the next time step is simply equal to that of the most recent observation at that sensor.

The first benchmark illustrates the effectiveness of our Gaussian process formalism that expresses correlations between different sensors. The second benchmark is a state-of-the-art alternative, that has previously been used for tracking environmental sensor data [Bertino et al. 2003]. Finally, the naïve third benchmark provides a lower bound on the performance of a prediction algorithm.

For the purposes of illustration, we will plot the sensor readings acquired by our algorithm (shown as markers), the mean and standard deviation of the GP prediction (shown as a solid line with plus or minus a single standard deviation shown as shading), and the true fine-grained sensor readings (shown as bold) that were downloaded directly from the sensor (rather than through the web site) after the event. Note that we present a limited number of sensors for reasons of space, but we use readings from all relevant sensors in order to perform inference.

Where appropriate, predictive quality will be evaluated using the standard root mean squared error (RMSE), defined as

$$\mathrm{RMSE}(\boldsymbol{m}, \boldsymbol{y}) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(m_i - y_i)^2}\,.$$

Here $\boldsymbol{m}$ is a vector of N estimates for the vector of variables $\boldsymbol{y}$. The estimates we use are the means for $\boldsymbol{y}$ conditioned on some set of observations $\boldsymbol{z}$ (as dependent on the application). We also provide the normed mean squared errors (NMSEs), defined as

$$\mathrm{NMSE}(\boldsymbol{m}, \boldsymbol{y}) = 10\log_{10}\frac{{}^{1}\!/_{N}\sum_{i=1}^{N}(m_i - y_i)^2}{{}^{1}\!/_{N}\left(\sum_{i=1}^{N} y_i^2\right) - \left({}^{1}\!/_{N}\sum_{i=1}^{N} y_i\right)^2}\,,$$

the ratio of the mean squared error and the variance of our data, expressed in units of decibels. The NMSE provides a measure of the accuracy in our predictions that is relatively insensitive to the scale of variation (and units) of the relevant variable.

It is commonly the case that after a significant volume of data, the majority of the weight will be concentrated at a single trial hyperparameter sample – the posterior distribution for the hyperparameter $p(\,\phi \mid \boldsymbol{z}_d,\, I\,)$ will be close to a delta function at that value. In this case, we will say, informally, that we have *learned* the hyperparameter to have that single value. Where this appears to be the case,

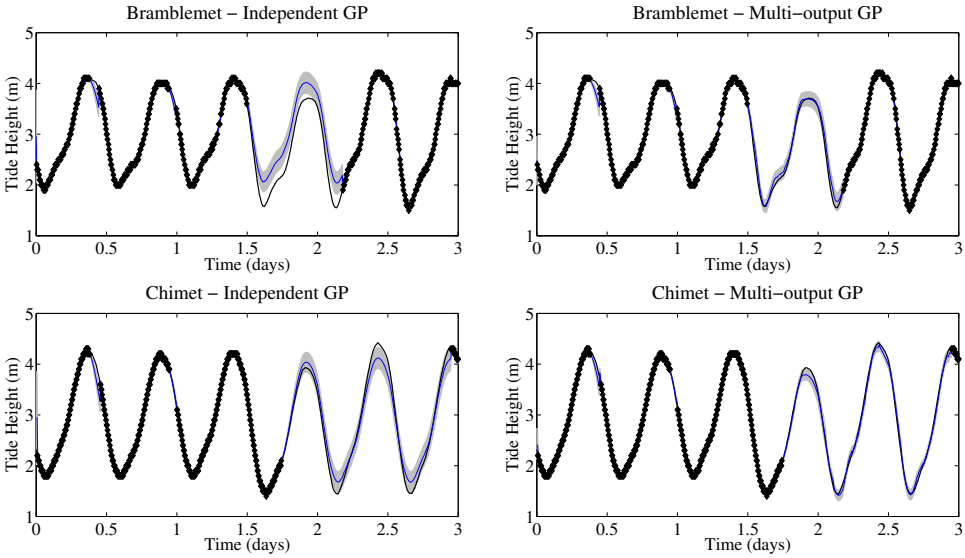

Fig. 5. Prediction and regression of tide height data for (a) independent and (b) multi-output Gaussian processes.

performing a local optimisation of the likelihood $p(\boldsymbol{z}_d \mid \phi, I)$ can help to find the point of highest posterior density.

In the subsections that follow we first evaluate our approach on tide height, air temperature and rounded air pressure data from the Bramblemet weather sensor network, and then discuss the prediction of maximum daily air temperature over the larger MIDAS land surface weather station network. In each case, we describe the covariance function used, and present quantitative comparisons of prediction quality. We then go on to discuss our approach to active data selection using the tide and wind speed sensors of the Bramblemet weather sensor network, and the air temperature sensors of the Wannangrat Alpine Observatory.

5.1 Prediction of Tide Heights

Figure 5 illustrates the efficacy our GP prediction for a tide height dataset. That is, at time t, figure 5 depicts the posterior distribution of the GP, conditioned on all observations prior to and inclusive of t. In order to manage the four outputs of our tide function (one for each sensor), we rewrite so that we have a single output and inputs t, time, and l, a sensor label. As such, our covariance function is of the form

$$K\big([t,\, l],\, [t',\, l']\big) = W(l,\, l')\,\Big(K_{\text{per-Mtn}}\big(t-\Delta_l, t'-\Delta_{l'}; h_P=1,\, w_P, \nu=\tfrac{5}{2}\big)+ K_{\text{Mtn}}\big(t-\Delta_l, t'-\Delta_{l'}; h_D,\, w_D, \nu=\tfrac{5}{2}\big)\Big), \quad (20)$$

where W is a covariance matrix parameterised using the spherical parameterisation (5). We used two different GP models: one in which the parameters of the spherical

Table I. Predictive performances for five-day Bramblemet tide height dataset.

Algorithm	RMSE (m)	NMSE (dB)
Naïve	7.5×10^{-1}	-2.1
Kalman filter	1.7×10^{-1}	-15.2
Independent GPs	8.7×10^{-2}	-20.3
Multi-output GP	3.8×10^{-2}	-27.6

decomposition were inferred from the data; and one for which the matrix S was taken as an identity matrix. In the former case, we consequently have a multi-output formalism, in the latter, independent GPs for each sensor.

Note that our covariance over time is the sum of a periodic term and a *disturbance* term. Both are of the Matérn form with $\nu = \frac{5}{2}$, (3) and (2) respectively. This form is a consequence of our expectation that the tides would be well modelled by the superposition of a simple periodic signal and an occasional disturbance signal due to exceptional conditions. Of course, for a better fit over the course of, say, a year, it would be possible to additionally incorporate longer-term drifts and periods.

The period w_P of the first term was unsurprisingly learnt as being about half a day, whereas for the disturbance term the time scale w_D was found to be about two and a half hours. Note that this latter result is concordant with our expectations for the time scales of the weather events we intend our disturbance term to model.

Our algorithm learned that all four sensors were very strongly correlated, with $S^{\mathrm{T}}S$ (where S is from (5)) containing elements all very close to one. W additionally gives an appropriate length scale for each sensor. Over this data set, the Chimet sensor was found to have a length scale of 1.4m, with the remainder possessing scales of close to 1m. Note that h_P and h_D in (20) are dimensionless ratios, serving as weightings between our various covariance terms. Hence we can set $h_D = 1$ without loss of generality, upon which we learn that h_P is around 0.2. Hence weather events were observed to have induced changes in tide height on the order of 20%.

We also make allowances for the prospect of relative latency amongst the sensors by incorporating delay variables, represented by the vector Δ. We found that the tide signals at the Cambermet and Chimet stations were delayed by about 10 minutes relative to the other two. This makes physical sense – the Bramblemet and Sotonmet stations are quite exposed to the ocean, while water has to run further through reasonably narrow channels before it reaches the Cambermet and Chimet stations.

Only the correlations amongst sensors and their individual delays were sampled over, all other hyperparameters for our covariance were learned by the GP prior to the depicted data set. Such hyperparameters were then specified with single samples at the learned values.

Note the performance of our multi-output GP formalism when the Bramblemet sensor drops out at $t = 1.45$ days. In this case, the independent GP quite reasonably predicts that the tide will repeat the same periodic signal it has observed in the past. However, the GP can achieve better results if it is allowed to benefit from the knowledge of the other sensors' readings during this interval of missing data. Thus, in the case of the multi-output GP, by $t = 1.45$ days, the GP has successfully

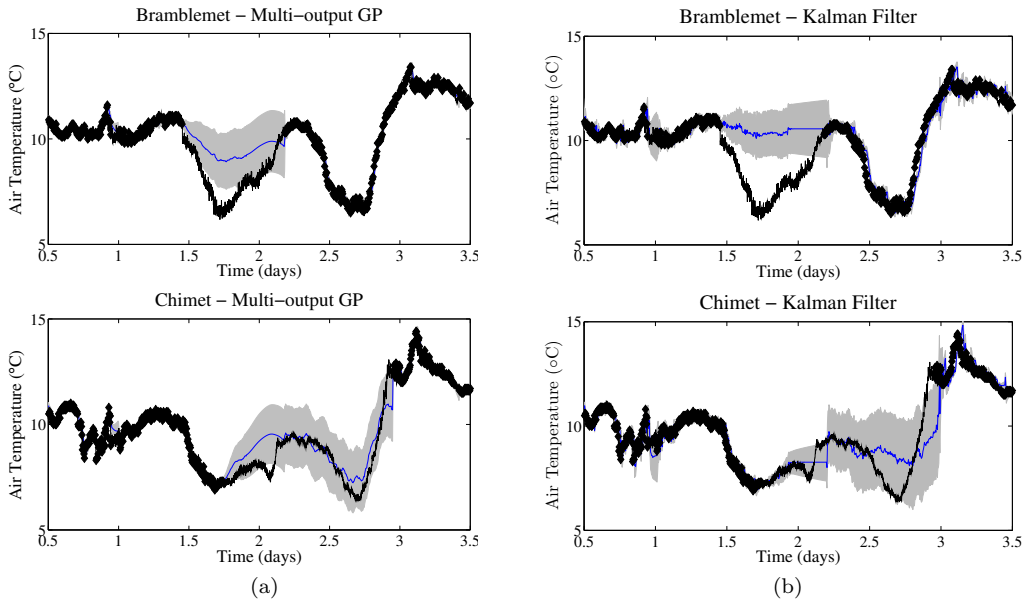

Fig. 6. One-step lookahead prediction of air temperature data for (a) a multi-output Gaussian process and (b) a Kalman filter.

determined that the sensors are all very strongly correlated. Hence, when it sees an unexpected low tide in the Chimet sensor data (caused by the strong Northerly wind), these correlations lead it to infer a similarly low tide in the Bramblemet reading. Hence, the multi-output GP produces significantly more accurate predictions during the missing data interval, with associated smaller error bars. Exactly the same effect is seen in the later predictions of the Chimet tide height, where the multi-output GP predictions use observations from the other sensors to better predict the high tide height at $t = 2.45$ days.

Note also that there are two brief intervals of missing data for all sensors just after both of the first two peak tides. During the second interval, the GP's predictions for the tide are notably better than for the first – the greater quantity of data it has observed allows it to produce more accurate predictions. With time, the GP is able to build successively better models for the series.

The predictive performances for our various algorithms over this dataset can be found in table I. Note that for the Kalman filter tested, the lookback p was taken as 16. Note that our multi-output GP which exploits correlations between the sensors, and the periodicity in each individual sensors' measurements, significantly outperforms the Kalman filter and the independent GP.

5.2 Prediction of Air Temperatures

We next tested on air temperature data. Figure 6 demonstrates our algorithm's 5 minute lookahead predictive performance, where 5 minutes was also the usual (although not exclusive) time increment from one datum to the next. That is, figure 6 depicts the posterior distribution at time t conditioned on all observations

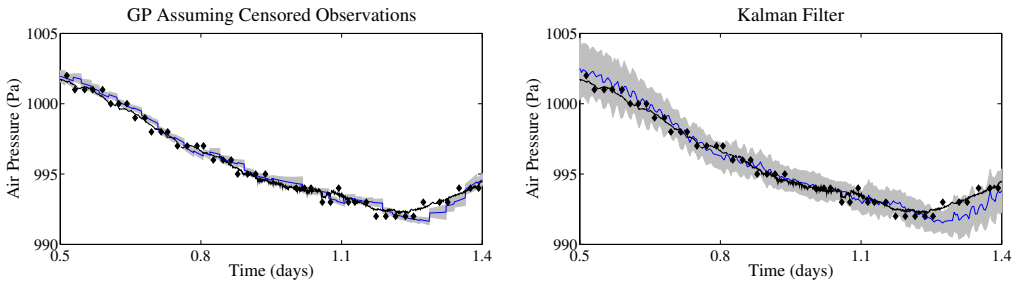

Fig. 7. Prediction and regression of air pressure data for (a) a Gaussian process employing censored observations and (b) a Kalman filter.

prior to and inclusive of $t - 5$ mins. Note that the Cambermet station does not record air temperature, and is hence discluded from consideration here. Taking the covariance function used for tide height data shown in equation (20) as a model, we assumed the covariance function

$$\begin{aligned} K\big([t,\, l],\, [t',\, l']\big) = W(l,\, l') \\ \Big(K_{\text{per-Mtn}}\big(t - \Delta_l, t' - \Delta_{l'}; h_P = 1,\, w_P, \nu = \tfrac{5}{2}\big) + \\ K_{\text{Mtn}}\big(t - \Delta_l, t' - \Delta_{l'}; h_T,\, w_T, \nu = \tfrac{5}{2}\big)\Big) + \\ \delta(l - l')\, K_{\text{Mtn}}\big(t, t'; h_D,\, w_D, \nu = \tfrac{3}{2}\big)\,, \end{aligned}$$

Our covariance over time consisted of three terms. The first was periodic, whose period w_P was learned as 1 day, describing nightly dips in temperature. The second term describes longer-term trends in temperature. Its time scale w_T was learned as 3 days, with a relative magnitude h_T just over twice that of the periodic term, h_P. From this we can induce that the periodic term represents only a relatively minor component of this model. Finally, the third term represents higher frequency disturbances, with a time scale learned as 4 hours. Note that this final term is not correlated amongst sensors as were the other two; these disturbances appear to be local to each sensor. The individual output scales for each sensor given by W were learned as 2°C, 6°C and 2.6°C respectively. These give the scales of the periodic fluctuations and longer-term drifts in the air temperature at each sensor. For comparison, the scale of the disturbances h_D were learned as 1°C. As for the tide heights example, only the sensor correlations and delays were sampled over for the depicted predictions.

The predictive performances for this dataset are shown in table II. The Kalman filter was tested with an order of $p = 15$. Note again that the multi-output GP outperforms the other benchmarks with the Kalman filter performing particularly badly, it significantly lagging the real temperature measurement.

Table II. Predictive performances for five-day Bramblemet air temperature dataset.

Algorithm	RMSE (°C)	NMSE (dB)
Naïve	0.84	-10.0
Kalman filter	0.98	-9.0
Independent GPs	0.82	-10.5
Multi-output GP	0.45	-15.4

5.3 Prediction of Air Pressures

Figure 7 demonstrates regression and prediction over the heavily rounded air pressure observations from the Bramblemet sensor alone. We used a GP model with the Matérn covariance of the non-periodic form shown in equation (2). We further made use of the methods described in Section 3.3 for managing the censored (rounded) observations that are all we possess for this data. Here we can demonstrate a dramatic improvement over Kalman filter prediction, as demonstrated in table III. Both the GP and Kalman filter have an order of 16; that is, they store only up to the 16 most recent observations (i.e., $p = 16$).

5.4 Prediction of MIDAS data

We now turn to a larger sensor network and demonstrate the suitability of our prediction algorithm for such networks. In particular, we apply our algorithm to data from January 1990 collected by the 670 sensors that comprised UK Met Office MIDAS land surface stations at the time. We arbitrarily selected maximum air temperature as the environmental variable of interest for our experiments, and aimed to perform one-step lookahead prediction. Specifically, we aim to predict the value of the next observation to be received by the network, which may be an observation from a day equal to that of the current observation, but from a sensor different to that most recently observed. The use of a model over both space and time is integral to success on this task.

The dataset comes equipped with latitude and longitude measurements for each sensor, which we can use to express the covariance between any pair of sensors. That is, we take the covariance

$$K\big([t,\, l],\, [t',\, l']\big) \,=\, K_{\mathrm{Mtn}}\big(l, l'; h,\, w_l, \nu \,=\, \tfrac{3}{2}\big)\, K_{\mathrm{Mtn}}\big(t, t'; h \,=\, 1,\, w_t, \nu \,=\, \tfrac{3}{2}\big)\,, \quad (21)$$

where for the covariance between l and l' we replace r with the great-circle distance between the positions of sensors l and l', divided by isotropic spatial scale hyperparameter w_l. Note that our approach here is an alternative to the arbitrary parameterisation presented in equation (5). This parameterisation requires a hyperparameter for every distinct pair of sensors and is hence impractical for larger numbers of sensors.

Results from our tests on this dataset are displayed in table IV. Note that the GP is easily able to outperform its naive competitor.

5.5 Active Data Selection of Tide Heights

We now demonstrate our active data selection algorithm. Using the fine-grained data (downloaded directly from the Bramblemet weather sensors), we can simulate

Table III. Predictive performances for one-day Bramblemet air pressure dataset. Note that with data from only one sensor, our multi-output GP formalism is not applicable.

Algorithm	RMSE (Pa)	NMSE (dB)
Naïve	3.61	-4.75
Kalman filter	3.29	-5.55
GP for censored data	0.43	-20.49

Table IV. Predictive performances for MIDAS maximum air temperature data.

Algorithm	RMSE (°C)	NMSE (dB)
Naïve	3.93	3.61
Basis functions	2.23	-1.32
Independent GPs	2.50	-0.31
Multi-output GP	1.53	-4.57

how our GP would have chosen its observations had it been in control. Results from the active selection of observations from all the four tide sensors are displayed in figure 8. Again, these plots depict dynamic choices; at time t, the GP must decide when next to observe, and from which sensor, given knowledge only of the observations recorded prior to t, in an attempt to maintain the uncertainty in tide height below 10cm. The covariance function used was that described in (20).

Consider first the case shown in figure 8(a), in which separate independent GPs are used to represent each sensor. Note that a large number of observations are taken initially as the dynamics of the sensor readings are learnt, followed by a low but constant rate of observation. In contrast, for the multi-output case shown in figure 8(b), the GP is allowed to explicitly represent correlations and delays between the sensors. As mentioned above, this data set is notable for the slight delay of the tide heights at the Chimet and Cambermet sensors relative to the Sotonmet and Bramblemet sensors, due to the nature of tidal flows in the area. Note that after an initial learning phase as the dynamics, correlations, and delays are inferred, the GP chooses to sample predominantly from the undelayed Sotonmet and Bramblemet sensors[5]. Despite no observations of the Chimet sensor being made within the time span plotted, the resulting predictions remain remarkably accurate. Consequently only 119 observations are required to keep the uncertainty below the specified tolerance, whereas 358 observations were required in the independent case. This represents another clear demonstration of how our prediction is able to benefit from the readings of multiple sensors.

5.6 Active Data Selection of Wind Speeds

Figure 9 shows similar results for the wind speed measurements from three of the four sensors (the Cambermet sensor being faulty during this period) where the goal

[5]The dynamics of the tide height at the Sotonmet sensor are more complex than the other sensors due to the existence of a 'young flood stand' and a 'double high tide' in Southampton. For this reason, the GP selects Sotonmet as the most informative sensor and samples it most often.

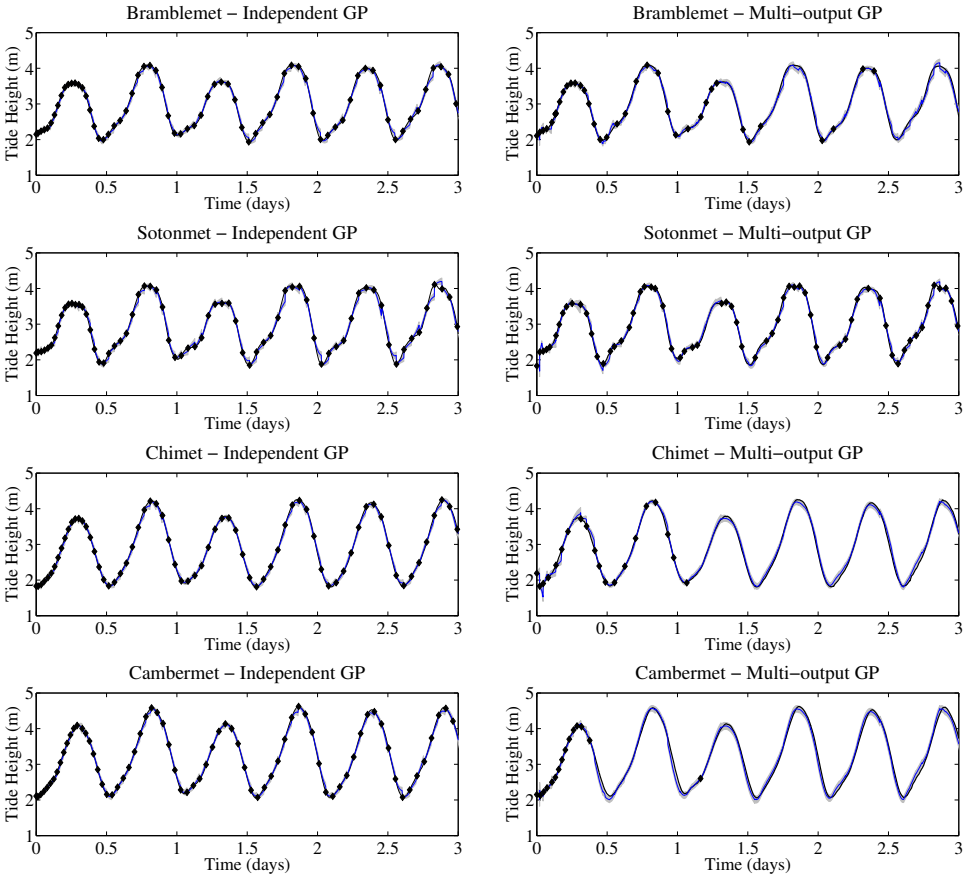

Fig. 8. Comparison of active sampling of tide data using (a) independent and (b) multi-output Gaussian processes.

was to maintain the uncertainty in wind speed below 1.5 knots. By analogy to the covariance function over tides (20), we used the following covariance function

$$K\big([t,\, l],\, [t',\, l']\big) = W(l,\, l')\, K_{\text{per-Mtn}}\big(t - \Delta_l, t' - \Delta_{l'}; h_P = 1,\, w_P, \nu = \tfrac{1}{2}\big) + \\ \delta(l - l')\, K_{\text{Mtn}}\big(t, t'; h_D,\, w_D, \nu = \tfrac{3}{2}\big)\,.$$

In this case, for purposes of clarity, the fine-grained data is not shown on the plot. Note that the measurement noise is much greater in this case, and this is reflected in the uncertainty in the GP predictions. Furthermore, note that while the Sotonmet and Chimet sensors exhibit a noticeable correlation, Bramblemet appears to be relatively uncorrelated with both. This observation is reflected in the sampling that the GP performs. The independent GPs sample the Bramblemet, Sotonmet and Chimet sensors 126, 120 and 121 times respectively, while over the same period, our multi-output GP samples the same sensors 115, 88 and 81 times. Our multi-

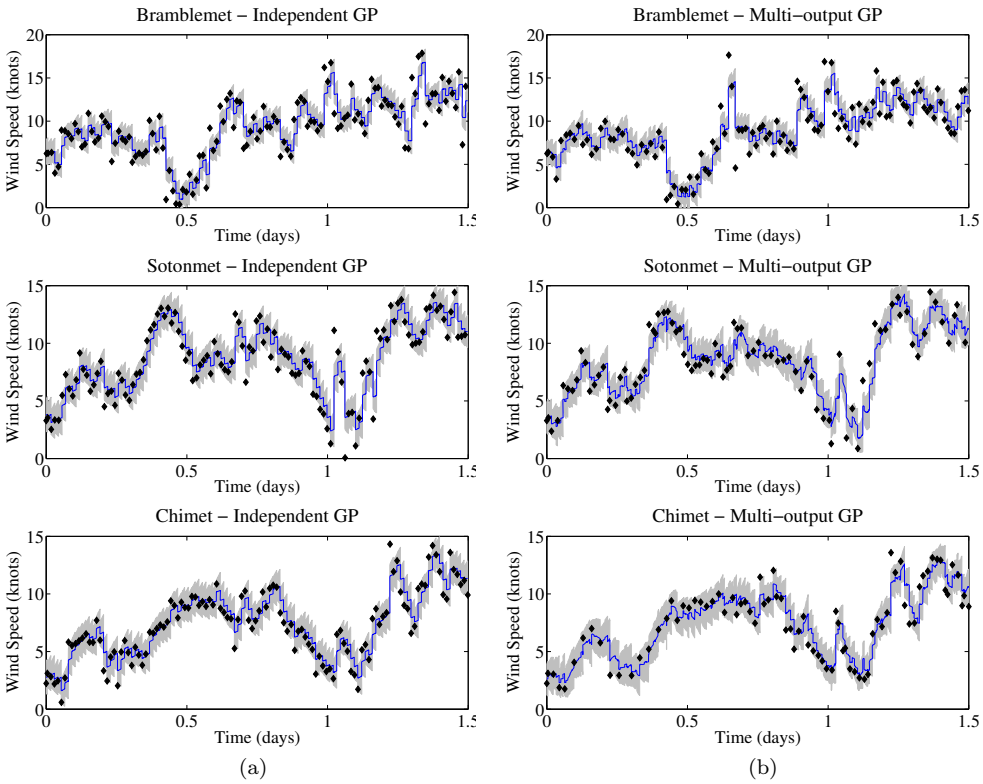

Fig. 9. Comparison of active sampling of wind speed using (a) independent and (b) multi-output Gaussian processes.

output GP learns on-line that the wind speed measurements of the Sotonmet and Chimet sensors are correlated, and then exploits this correlation in order to reduce the number of times that these sensors are sampled (inferring the wind speed at one location from observations of another). However, there is little or no correlation between the Bramblemet sensor and the other sensors, and thus, our multi-output GP samples Bramblemet almost as often as the independent GPs.

5.7 Active Data Selection of Air Temperature

To investigate the practicality of our active data selection algorithm on larger sensor networks, we apply it to data from 16 air temperature sensors from the Wannengrat Alpine Observatory. As in the case of the MIDAS data described above we use the latitude and longitude co-ordinates of these sensors within the covariance function shown in equation (21).

Figure 10 shows the detailed predictions and samples for five of the sensors, and a summary plot showing the sample times of all 16 sensors. Note that the sensor sampling rate is relatively low at first, however, as the volatile fluctuations in temperature begin to occur at about $t = 0.7$ days, the sampling frequency automatically increases to ensure that the prediction uncertainty is maintained

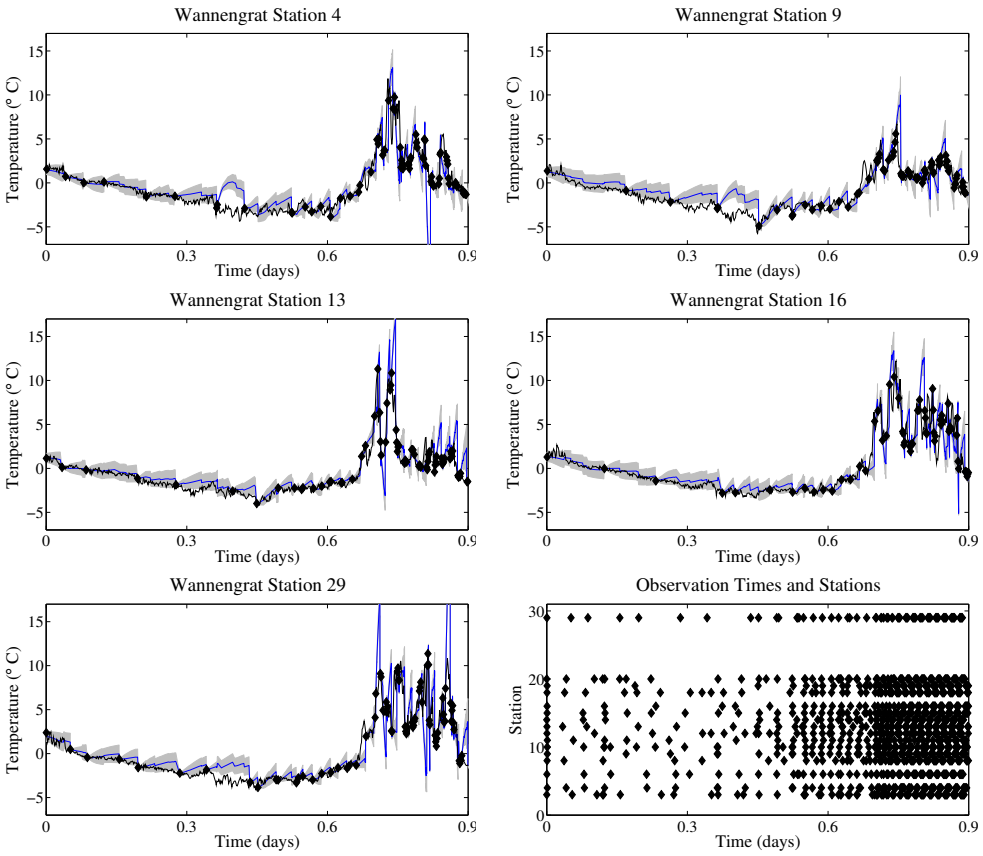

Fig. 10. Active sampling of ambient temperatures at 16 Wannengrat sensor stations.

below the specified threshold.

6. COMPUTATION TIME

As described earlier, a key requirement of our algorithm is computational efficiency, in order that it can be used to represent multiple correlated sensors, and hence, used for real-time information processing. Here we consider the computation times involved in producing the results presented in the previous section. We should note at this point that we are not comparing the computation time of our approach against the Kalman filter. The computation performed by the Kalman filter is far less intensive than that of the Gaussian process (in our empirical evaluations, the update time of the Kalman filter was typically < 0.01 seconds). However, as demonstrated in the previous section, the Kalman filter is unable to represent the complex correlations and periodicities seen in the sensor data, and thus, it provides significantly worse predictions than our approach.

Rather, in this section, we are evaluating the computation time required in order to update the Gaussian process as a new observation is received. This computation

Table V. The required computation time (seconds) per GP update, with (a) and without (b) our efficient update rules, over N the number of stored data points and η the number of hyperparameter samples.

	N		
η	10	100	500
1	< 0.01	< 0.01	< 0.01
10	0.01	0.01	0.08
100	0.11	0.13	0.74
1000	1.30	1.39	4.51

(a)

	N		
η	10	100	500
1	< 0.01	< 0.01	0.06
10	0.01	0.02	0.34
100	0.09	0.17	3.32
1000	0.92	1.71	17.34

(b)

time represents the cost of computing the Cholesky factor $\mathbf{R}$ of $\mathbf{V}$, the associated data-dependent term $\mathbf{R}(\boldsymbol{x}_d, \boldsymbol{x}_d)^{-1}(\boldsymbol{y}_d - \boldsymbol{\mu}(\boldsymbol{x}_d))$ in equations (7) and (8), and the hyperparameter sample likelihoods and subsequently the weights of equation (16). Once these terms have been calculated, by either method, making predictions at any point in time is extremely fast, requiring simply the adding of another element to $\boldsymbol{x}_\star$. The conventional approach to performing this update would be to simply recompute the relevant quantities from scratch upon receipt of new data (using the highly optimised LAPACK routines within Matlab - see `http://www.netlib.org/lapack/` for more details of this linear algebra package). In contrast, in our formalism, we make use of the computationally efficient online rules described in Section 3.4 to perform an online update of previously calculated results.

Thus, against this background, Table V shows the computation time required using MATLAB on an Intel Core i7 Processor 860 (2.80GHz, 8MB) with 8GB of RAM, with displayed results the average of 10 trials[6]. In more detail, Table V(a) the results when our computationally efficient online rules are used, and Table V(b) shows the results when the conventional approach is used.

Note that with our efficient updates, we expect the cost of computation to grow as $O(N^2)$ in the number of stored data points. Our proposed algorithm will automatically determine the quantity of data to store in order to achieve the desired level of accuracy. In the problems we have studied, a few hundred points were typically sufficient. The largest number we required was 1500, for the MIDAS air temperature data. Note also that the cost of computing equation (19) will grow in the cube of the number of samples in each hyperparameter. However, we consider only a fixed set of samples in each hyperparameter, and thus, equation (19) need only be computed once, off-line. In this case, our on-line costs are limited by the multiplication of that term by the likelihoods $\boldsymbol{r}_s$ to give the weights of equation (16), and this only grows as $O(\eta^2)$. Furthermore, note that this cost is independent of how the η samples are distributed amongst the hyperparameters. For completeness, Table VI shows the number of data points and the number of hyperparameter samples used for each of the empirical evaluations.

In practice, we see that our efficient updates result in substantial improvements in

[6]Note that these computations would typically be performed at the base station or data portal, and not on the sensors themselves, so reasonable computational resources can be assumed.

Table VI. The numbers of stored data points, N, and hyperparameter samples, η, used for the various datasets presented in this paper.

Dataset	N	η
Bramblemet Tide Heights	200	1620
Bramblemet Air Temperatures	500	972
Bramblemet Air Pressures	16	125
MIDAS Air Temperature	1500	105
Bramblemet Wind Speeds	454	675
Wannengrat Air Temperature	500	125

computational speed once the quantity of data N exceeds the threshold required to offset the small computational overheads introduced by using our approach instead of the already highly optimised LAPACK routines. For the largest cases investigated here we see an approximate four-fold increase in computation speed. For our weather datasets, samples are taken no more frequently than once a minute. Thus all of the information processing that we have described in this paper can easily be performed in real-time, and the increase in computational speed seen in our approach can be used to increase the number of data points considered or the number of hyperparameters samples used; both of which will likely increase the predictive power of the Gaussian process.

7. RELATED WORK

Gaussian process regression has a long history of use within geophysics and geospatial statistics (where the process is known as kriging [Cressie 1991]), but has only recently been applied within sensor networks. For example, Gaussian processes have been used to represent spatial correlations between sensors so that the near-optimal sensor placement (in terms of mutual information) can be determined, for modelling wireless propagation between sensor nodes subject to censored observations of received signal strength [Ertin 2007], and in the form of multi-variate Gaussian distributions to represent correlations between different sensors and sensor types for energy efficient querying of a sensor network [Deshpande et al. 2004]. They have also been used to represent temporal correlations between the readings from a single sensor in order to perform adaptive sampling — maximising the information gain subject to a sampling constraint [Kho et al. 2009].

Our work differs in that we use GPs to represent temporal correlations, and represent correlations and delays between sensors with additional hyperparameters. It is thus closely related to other work using GPs to perform regression over multiple responses [Boyle and Frean 2005; Teh et al. 2005]. However, our focus is to derive a computationally efficient algorithm, and thus, we use a number of novel computational techniques to allow the re-use of previous calculations as new sensor observations are made. We additionally use a novel Bayesian Monte Carlo technique to marginalise the hyperparameters that describe the correlations and delays between sensors. Finally, we use the variance of the GP's predictions in order to perform active data selection. Likewise, our work differs from that previously using censored sensor readings within a GP framework [Ertin 2007], since our work pro-

poses a principled Bayesian Monte Carlo method for adapting our models to the data, where Ertin's model assumes that the hyperparameters are known a priori, and hence, does not consider how likelihoods should be computed in this context. Furthermore, in our work, Monte Carlo techniques are also used to evaluate our other integrals, rather than taking Laplace approximations, allowing more accurate representation of the correlation amongst censored readings.

Finally, our approach has several advantages relative to sequential state-space models such as the Kalman filter [Girard et al. 2003; Jazwinski 1970] which have also been applied in environmental settings for tracking sensor readings [Bertino et al. 2003]. Firstly, these state-space models require the discretisation of the time input, representing a discarding of potentially valuable information. Secondly, their sequential nature means they must necessarily perform difficult iterations in order to manage missing or late data, or to produce long-range forecasts. In our GP approach, what observations we have are readily managed, regardless of when they were made. Equally, the computation cost of all our predictions is identical, irrespective of the time or place we wish to make them about. Finally, a sequential framework requires an explicit specification of a transition model. In our approach, we are able to learn a model from data even if our prior knowledge is negligible, and the benefits of our approach are empirically supported by the results presented in Section 5.

8. CONCLUSIONS

In this paper we addressed the need for algorithms capable of performing real-time information processing of sensor network data, and we presented a novel computationally efficient formalism of a multi-output Gaussian process. Using weather data collected from three sensor networks, we demonstrated that this formalism allows an end-user to make effective use of sensor data even in the presence of network outages and sensor failures (recovering missing data by making predictions based on previous sensor readings and the current readings of sensors that are functioning), and to automatically determine the sampling rate of each individual sensor in order to ensure that the uncertainty in the environmental parameter being measured stays within a pre-specified limit. Furthermore, we showed that our formalism that efficiently re-uses previous computations by following an online update procedure as new data sequentially arrives results a significant increase in computation speed compared to the highly optimised LAPACK linear algebra package.

Our future work in this area consists of three areas. First, we would like to investigate the use of schemes more sophisticated than the maximum likelihood approach for the movement of the locations of our set of hyperparameter samples. As the posterior distributions of these hyperparameters become more sharply peaked, we might aim to reduce the number of samples to further increase the computational efficiency of our algorithm.

Second, we intend to investigate the use of correlations between different sensor types (rather than between different sensors of the same type as presented here) to perform regression and prediction within our weather sensor network. Our formalism is in no way restricted to identical sensors, and it is expected that many sensors will exhibit correlations in this setting. For example, both wind speed and wind

direction, and air temperature and air pressure are likely to exhibit correlations as weather fronts move over the sensor network.

Finally, we would like to use the probabilistic model that the GP builds to automatically handle faulty and unreliable sensors within the network. Note that we do not wish to simply detect these faulty sensors., but would like to simultaneously perform both detection and prediction, despite the presence of these failures. Our preliminary work is this area indicates that nonstationary covariance functions that model such changes can be introduced to the formalism described here to achieve this, and have already been shown to work effectively on real-world sensor data derived from the sensor networks described in this paper [Garnett et al. 2009].

Acknowledgments

This research was undertaken as part of the ALADDIN (Autonomous Learning Agents for Decentralised Data and Information Networks) project, jointly funded by a BAE Systems and EPSRC strategic partnership (EP/C548051/1), and was also part of the Systems Engineering for Autonomous Systems (SEAS) Defence Technology Centre established by the UK Ministry of Defence. We would like to thank B. Blaydes of the Bramblemet/Chimet Support Group, and W. Heaps of Associated British Ports (ABP) for allowing us access to the weather sensor network, hosting our RDF data on the sensor web sites, and for providing raw sensor data as required. We also thank the UK Met office for the use of the MIDAS dataset.

REFERENCES

Abrahamsen, P. 1997. A review of Gaussian random fields and correlation functions. Tech. Rep. 917, Norwegian Computing Center, Box 114, Blindern, N-0314 Oslo, Norway. 2nd edition.

Barnaghi, P., Meissner, S., Presser, P., and Moessner, K. 2009. Sense and sens'ability: Semantic data modelling for sensor networks. In *Proceedings of the ICT Mobile and Wireless Communications Summit*. Santander, Spain.

Bertino, L., Evensen, G., and Vackernagel, H. 2003. Sequential data assimilation techniques in oceanography. *International Statistical Review* 71, 223–242.

Boyle, P. and Frean, M. 2005. Dependent Gaussian processes. In *Advances in Neural Information Processing Systems 17*. The MIT Press, 217–224.

British Atmospheric Data Centre. 2009. UK Meteorological Office MIDAS Land Surface Stations data (1853-current). Available from: `http://badc.nerc.ac.uk/data/ukmo-midas`.

Cressie, N. A. C. 1991. *Statistics for spatial data*. John Wiley & Sons.

Deshpande, A., Guestrin, C., Madden, S., Hellerstein, J., and Hong, W. 2004. Model-driven data acquisition in sensor networks. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases (VLDB 2004)*. Toronto, Canada, 588–599.

Ertin, E. 2007. Gaussian Process Models for Censored Sensor Readings. In *Proceedings of the 14th IEEE/SP Workshop on Statistical Signal Processing*. Madison, Wisconsin, USA, 665–669.

Fuentes, M., Chaudhuri, A., and Holland, D. H. 2007. Bayesian entropy for spatial sampling design of environmental data. *Environmental and Ecological Statistics* 14, 323–340.

Garnett, R., Osborne, M. A., and Roberts, S. J. 2009. Sequential bayesian prediction in the presence of changepoints. In *Proceedings of the 26th International Conference on Machine Learning*. Montreal, Canada.

Genz, A. 1992. Numerical Computation of Multivariate Normal Probabilities. *Journal of Computational and Graphical Statistics 1,* 2, 141–149.

Girard, A., Rasmussen, C., Candela, J., and Murray-Smith, R. 2003. Gaussian process priors with uncertain inputs – application to multiple-step ahead time series forecasting. In *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 545–552.

Hart, J. K. and Martinez, K. 2006. Environmental Sensor Networks: A revolution in the earth system science? *Earth-Science Reviews 78*, 177–191.

Jazwinski, A. 1970. *Stochastic processes and filtering theory*. Academic Press New York.

Kho, J., Rogers, A., and Jennings, N. R. 2009. Decentralized control of adaptive sampling in wireless sensor networks. *ACM Transactions on Sensor Networks 5,* 3, 1–35.

Krause, A., Guestrin, C., Gupta, A., and Kleinberg, J. 2006. Near-optimal sensor placements: Maximizing information while minimizing communication cost. In *Proceedings of the Fifth International Conference on Information Processing in Sensor Networks*. Nashville, Tennessee, USA, 2–10.

Lassila, O. and Swick, R. R. 1999. Resource description framework (rdf) model and syntax specification. Available at `http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/`.

MacKay, D. J. C. 2002. *Information Theory, Inference & Learning Algorithms*. Cambridge University Press.

O'Hagan, A. 1987. Monte Carlo is fundamentally unsound. *The Statistician 36*, 247–249.

Padhy, P., Dash, R. K., Martinez, K., and Jennings, N. R. 2010. A utility-based adaptive sensing and multi-hop communication protocol for wireless sensor networks. *ACM Transactions on Sensor Networks 6,* 3. In print.

Pinheiro, J. and Bates, D. 1996. Unconstrained parameterizations for variance-covariance matrices. *Statistics and Computing 6*, 289–296.

Rasmussen, C. E. and Ghahramani, Z. 2003. Bayesian Monte Carlo. In *Advances in Neural Information Processing Systems 15*. The MIT Press, 489–496.

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

Sasena, M. J. 2002. Flexibility and Efficiency Enhancements for Constrained Global Design Optimization with Kriging Approximations. Ph.D. thesis, University of Michigan.

Shumway, R. and Stoffer, D. 2000. *Time Series Analysis and its applications*. Springer.

Stein, M. 2005. Space-Time Covariance Functions. *Journal of the American Statistical Association 100,* 469, 310–322.

Teh, Y. W., Seeger, M., and Jordan, M. I. 2005. Semiparametric latent factor models. In *Proceedings of the Conference on Artificial Intelligence and Statistics*. Barbados, 333–340.

The MathWorks. 2007. MATLAB R2007a. Natick, MA.

A. APPENDIX

A.1 Cholesky Factor Update

We have a positive definite matrix, represented in block form as $\begin{bmatrix} V_{1,1} & V_{1,3} \\ V_{1,3}^{\mathrm{T}} & V_{3,3} \end{bmatrix}$ and its Cholesky factor, $\begin{bmatrix} R_{1,1} & R_{1,3} \\ 0 & R_{3,3} \end{bmatrix}$. Given a new positive definite matrix, which differs from the old only in the insertion of some new rows and columns, $\begin{bmatrix} V_{1,1} & V_{1,2} & V_{1,3} \\ V_{1,2}^{\mathrm{T}} & V_{2,2} & V_{2,3} \\ V_{1,3}^{\mathrm{T}} & V_{2,3}^{\mathrm{T}} & V_{3,3} \end{bmatrix}$, we wish to efficiently determine its Cholesky factor, $\begin{bmatrix} S_{1,1} & S_{1,2} & S_{1,3} \\ 0 & S_{2,2} & S_{2,3} \\ 0 & 0 & S_{3,3} \end{bmatrix}$. For $\mathbf{A}$ triangular, we define $\boldsymbol{x} = \mathbf{A} \setminus \boldsymbol{b}$ as the solution to the equations $\mathbf{A}\,\boldsymbol{x} = \boldsymbol{b}$ as found by the use of backwards or

forwards substitution. The following rules are readily obtained

$$S_{1,1} = R_{1,1} \tag{22}$$
$$S_{1,2} = R_{1,1}^{\mathrm{T}} \setminus V_{1,2} \tag{23}$$
$$S_{1,3} = R_{1,3} \tag{24}$$
$$S_{2,2} = \mathrm{chol}(V_{2,2} - S_{1,2}^{\mathrm{T}} S_{1,2}) \tag{25}$$
$$S_{2,3} = S_{2,2}^{\mathrm{T}} \setminus (V_{2,3} - S_{1,2}^{\mathrm{T}} S_{1,3}) \tag{26}$$
$$S_{3,3} = \mathrm{chol}(R_{3,3}^{\mathrm{T}} R_{3,3} - S_{2,3}^{\mathrm{T}} S_{2,3})\,. \tag{27}$$

By setting the appropriate row and column dimensions (to zero if necessary), this allows us to efficiently determine the Cholesky factor given the insertion of rows and columns in any position.

A.2 Data Term Update

We have all terms defined in Section A.1, in addition to the data

$$\begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \end{bmatrix}$$

and the product

$$\begin{bmatrix} C_1 \\ C_3 \end{bmatrix} \triangleq \begin{bmatrix} R_{1,1} & R_{1,3} \\ 0 & R_{3,3} \end{bmatrix}^{-\mathrm{T}} \begin{bmatrix} Y_1 \\ Y_3 \end{bmatrix}.$$

To efficiently determine the updated product

$$\begin{bmatrix} D_1 \\ D_2 \\ D_3 \end{bmatrix} \triangleq \begin{bmatrix} S_{1,1} & S_{1,2} & S_{1,3} \\ 0 & S_{2,2} & S_{2,3} \\ 0 & 0 & S_{3,3} \end{bmatrix}^{-\mathrm{T}} \begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \end{bmatrix},$$

we have

$$D_1 = C_1 \tag{28}$$
$$D_2 = S_{2,2}^{-\mathrm{T}} \left(Y_2 - S_{1,2}^{\mathrm{T}}\, C_1\right) \tag{29}$$
$$D_3 = S_{3,3}^{-\mathrm{T}} \left(R_{3,3}^{\mathrm{T}}\, C_3 - S_{2,3}^{\mathrm{T}}\, D_2\right). \tag{30}$$

A.3 Log-Gaussian Update

We have all terms defined in Sections A.1 and A.2, in addition to

$$\begin{aligned} K &= \log \mathrm{N}\left(\begin{bmatrix} Y_1 \\ Y_3 \end{bmatrix}; \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} V_{1,1} & V_{1,3} \\ V_{1,3}^{\mathrm{T}} & V_{3,3} \end{bmatrix} \right) \\ &= -\frac{1}{2} \operatorname{tr} \log(\sqrt{2\pi} R_{1,1}) - \frac{1}{2} \operatorname{tr} \log(\sqrt{2\pi} R_{3,3}) - \frac{1}{2} C_1^{\mathrm{T}}\, C_1 - \frac{1}{2} C_3^{\mathrm{T}}\, C_3\,. \end{aligned}$$

We can then calculate the updated term

$$\begin{aligned} L &= \log \mathrm{N}\left(\begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \end{bmatrix} ; \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} , \begin{bmatrix} V_{1,1} & V_{1,2} & V_{1,3} \\ V_{1,2}^{\mathrm{T}} & V_{2,2} & V_{2,3} \\ V_{1,3}^{\mathrm{T}} & V_{2,3}^{\mathrm{T}} & V_{3,3} \end{bmatrix} \right) \\ &= -\frac{1}{2}\operatorname{tr}\log(\sqrt{2\pi}S_{1,1}) - \frac{1}{2}\operatorname{tr}\log(\sqrt{2\pi}S_{2,2}) - \frac{1}{2}\operatorname{tr}\log(\sqrt{2\pi}S_{3,3}) \\ &\quad - \frac{1}{2}D_1^{\mathrm{T}}\, D_1 - \frac{1}{2}D_2^{\mathrm{T}}\, D_2 - \frac{1}{2}D_3^{\mathrm{T}}\, D_3 \\ &= K + \frac{1}{2}\operatorname{tr}\log(\sqrt{2\pi}R_{3,3}) - \frac{1}{2}\operatorname{tr}\log(\sqrt{2\pi}S_{2,2}) - \frac{1}{2}\operatorname{tr}\log(\sqrt{2\pi}S_{3,3}) \\ &\quad + \frac{1}{2}C_3^{\mathrm{T}}\, C_3 - \frac{1}{2}D_2^{\mathrm{T}}\, D_2 - \frac{1}{2}D_3^{\mathrm{T}}\, D_3 \end{aligned} \tag{31}$$

A.4 Cholesky Factor Downdate

We have a positive definite matrix, represented in block form as $\begin{bmatrix} V_{1,1} & V_{1,2} & V_{1,3} \\ V_{1,2}^{\mathrm{T}} & V_{2,2} & V_{2,3} \\ V_{1,3}^{\mathrm{T}} & V_{2,3}^{\mathrm{T}} & V_{3,3} \end{bmatrix}$ and its Cholesky factor, $\begin{bmatrix} S_{1,1} & S_{1,2} & S_{1,3} \\ 0 & S_{2,2} & S_{2,3} \\ 0 & 0 & S_{3,3} \end{bmatrix}$. Given a new positive definite matrix, which differs from the old only in the deletion of some new rows and columns, $\begin{bmatrix} V_{1,1} & V_{1,3} \\ V_{1,3}^{\mathrm{T}} & V_{3,3} \end{bmatrix}$, we wish to efficiently determine its Cholesky factor $\begin{bmatrix} R_{1,1} & R_{1,3} \\ 0 & R_{3,3} \end{bmatrix}$. The following rules are readily obtained

$$R_{1,1} = S_{1,1} \tag{32}$$

$$R_{1,3} = S_{1,3} \tag{33}$$

$$R_{3,3} = \operatorname{chol}(S_{2,3}^{\mathrm{T}} S_{2,3} + S_{3,3}^{\mathrm{T}} S_{3,3})\,. \tag{34}$$

Note that the special structure of equation (34) can be exploited for the efficient resolution of the required Cholesky operation, as, for example, in the MATLAB function cholupdate [The MathWorks 2007]. By setting the appropriate row and column dimensions (to zero if necessary), this allows us to efficiently determine the Cholesky factor given the deletion of rows and columns in any position.