VARIATIONAL GAUSSIAN PROCESS FOR OPTIMAL SENSOR PLACEMENT

Gabor Tajnafoi, London, Rossella Arcucci, London, Laetitia Mottet, London, Carolanne Vouriot, London, Miguel Molina-Solana, Granada, Christopher Pain, London, Yi-Ke Guo, London

(Received November 15, 2019)

Abstract. Sensor placement is an optimisation problem that has recently gained great relevance. In order to achieve accurate online updates of a predictive model, sensors are used to provide observations. When sensors location is optimally selected, the predictive model can greatly reduce its internal errors. A greedy-selection algorithm is used for locating these optimal spatial locations from a numerical embedded space. A novel architecture for solving this big data problem is proposed, relying on a Variational Gaussian Process. The generalisation of the model is further improved via the preconditioning of its inputs: Masked Autoregressive Flows are implemented to learn non-linear, invertible transformations of the conditionally modelled spatial features. Finally, a global optimisation strategy extending the Mutual Information-based optimisation and fine-tuning of the selected optimal location is proposed. The methodology is parallelised to speed-up the computational time, making these tools very fast despite the high complexity associated with both spatial modelling and placement tasks. The model is applied to a real three-dimensional test case considering a room within the Clarence Centre building located in Elephant and Castle, London, UK.

Keywords: Sensor placement, Variational Gaussian Processes, Mutual Information

MSC 2010: 65Z05, 68T99

1. Introduction

Indoor Air Quality (IAQ) impacts health, comfort and quality of life [1], and three basic strategies have been proposed to improve it: control of pollution sources, use of natural/mechanical ventilation, and cleaning of air. In the building context, the management and development of smart monitoring tools can support an adequate

IAQ within them (e.g. by automatically opening windows or starting a mechanical air cleaning system). Sensors coupled with indoor pollutant forecasting models can tackle bad IAQ by implementing one of the previously-cited strategies before the indoor pollutant concentration reaches dangerous and adverse levels.

In order to achieve accurate, online updates of the predictive model, sensors can be used to provide observations. Spatio-temporal models such as Data Assimilation (DA) provide online learning and forecasting of sensor observations by means of updating the model's internal view through the incorporation of collected data [2, 3]. In this context, sensor positioning has gained relevance [4, 5] as it is crucial to ensure a good quality and usefulness of monitored data. Optimal sensor positioning tools pinpoint the discrete spatial locations that possess most conditional information of all other spatial points, thus improving the predictive accuracy of prediction models [6]. Hence, sensor placement can be seen as an optimisation problem [4, 5].

Early attempts on sensor placement used geometric approaches, supported by the assumption that sensors measure spatial features with a fixed sensing radius [7]. This geometric approach does not take into account the non-linear dynamic behaviour of air motion so, in order to tackle this problem, parametric models [8], non-parametric Gaussian Process (GP) [9] and ensemble Kalman-filters [10] approaches were subsequently proposed.

The main work on this field implements sensor placement using a GP in a 2D space [5]. The time complexity of the placement algorithm is $O(N^4)$, where N denotes the side of the computational domain. The GP is trained on data collected from fixed sensors located in a room, then N is relatively small. The placement algorithm only select the best sensors in the set already provided.

In this work, for the first time, a sensor placement model is developed for a 3D domain representing a real case scenario. Also, the model uses temporal sequences of data from fluid dynamic simulations facing then a big data problem. In our case, N is on an scale such that the use of a GP is unfeasible. To address this problem we developed a combination of deep learning, probabilistic frameworks and variational methods, reducing the complexity associated with training, inference and optimisation and thus, enabling us to achieve optimal placement results in a real case scenario. The complexity of our model is, in fact, $O(klM^4)$, where k is the number of sensors, l is the number of iteration needed to optimise the position and M is such that $M << N$.

The rest of this paper is organized as follows: next section introduces the background to this work and our main contributions. Section 3 presents the mathematical formulation of our proposed model, Variational Gaussian Process optimal sensor placement (VGPosp). Section 4 describes the direct application of VGPosp on a real indoor environment. The manuscript ends with some conclusions and further work.

2. Background and main contributions

Non-parametric models consist in learning a Gaussian Process (GP) associated with the phenomenology considered (e.g. pollution levels in indoor environments). In general, GPs are highly appropriate to study environmental problems as they allow for learning complex, high-dimensional correlations with uncertainty quantification. Indeed, non-parametric expressiveness is an advantage over parametric models that are more prone to the curse of dimensionality [11]. A GP, sometimes refered as the Bayesian interpretation of neural networks, is fully determined by only two parameters, namely the mean-function and covariance-function, regardless of their dimensionality [12]. Three main GP methodologies can be identified: the Traditional GP [4, 5], the Sparse GP [13] and the Variational GP [14, 15].

Traditional GP is a stochastic process based on prior distribution over functions and it has been successfully applied for indoor optimal sensor positioning [4, 5]. However, Traditional GP suffers from the high complexities associated with spatial modelling $O((mN)^3)$, where N denotes the size of input sensor potential locations and m the number of physical variables, which explains why the work presented in [4, 5] was only carried out in two dimensions.

Sparse Variational Process (SGP) tackles the inconvenient $O((mN)^3)$ computational complexity associated with Traditional GP [16]. This method constructs an approximation based on a small subset of size $\hat{N}$, namely inducing points. This optimisation results in a reduced complexity $O((mN)\hat{N}^2)$, enabling the scalability of training data-points from the previous limit of a few thousand to the range of millions [17]. In general, sparsity can be achieved by working on a low-rank representation of the full kernel [18]. The key idea is to approximate the prior or modify the likelihood function, thus creating a model selection problem solving the optimisation for the approximation of the truth [13]. However, the main criticism to SGPs is that they learn unknown hyperparameters, potentially leading them to underestimate variance and thus over-fitting [14, 15].

Alternatively, Variational Gaussian Process (VGP), a variational method for SGP, was developed [14, 15] to deal with the approximation of model components that

are hard to compute. Inducing points are variational parameters selected by minimising the Kullback-Leibler (KL) divergence [14, 15]. The kernel hyperparameters and inducing points are jointly optimised by maximising a lower bound (Evidence Lower BOund (ELBO)) of variational distribution over the functions latent values [14, 15, 19]. The key innovation, is that the likelihood and the GP prior are not modified, separating the model and the inference. The variational posterior iteratively approaches the true posterior.

The datasets used to train a GP usually comprises data coming from monitoring sensors during extensive field experiment periods. This training dataset usually suffer from being non-Gaussian distributed, rendering it unusable for GP learning [20]. In this regards, several methodologies to precondition, normalise and render the training dataset Gaussian can be mentioned: Variational Autoencoder [21], Autoregressive Flows [22], Normalising Flows [23], Masked Autoencoder for Distribution Estimation (MADE) [24], and Masked Autoregressive Flows (MAF) [20]. The MAF approach is a stack of MADE networks [24, 20] and has proved its competitiveness over the other methodologies in terms of accuracy [20].

Even if VGP can deal with non-Gaussian distributed data, the preconditioning phase of learning non-linear, invertible transformations between the conditional input distributions and output Gaussian family of distributions enables greater generalisation of the learned spatial model.

At this stage, the actual sensors placement problem can then be addressed using the trained GP. Indeed, the placement algorithm is solved in an embedded space that is predicted by a GP [14, 15]. In other words, the GP serves as a numerical setting for the optimisation problem: conditional predictions are used to generate the covariance matrix taken as input by the placement algorithm. The complexity associated with the placement task is $O(N^4)$, where N denotes the size of input sensor potential locations.

Mutual Information (MI) [4, 5] and minimum cross entropy [25, 26] are some of the metrics traditionally used. The use of minimum cross entropy tends to maximise the distance between sensors. In indoor environment problems, this results in having sensors located near the boundary of the domain, i.e. near the walls, thus loosing information monitored [26, 5]. One the other hand, information gain or Mutual Information [4] shifts the amount of information captured by a single random variable to the information each random variable has of the other unobserved one. More specifically, considering a finite set of possible placement locations, by maximising the objective metric, it evaluates how well a given smaller subset of sensor locations describes the values of the unselected other locations. This paper considers the

optimisation problem of sensor placement in indoor environments by separating the problem into the learning of a spatial model, i.e. Gaussian Process training, and the optimisation algorithm itself, i.e. optimal sensor placement. It is demonstrated that with a combination of deep learning, probabilistic frameworks and variational methods, the complexity associated with training, inference and optimisation can be significantly reduced in order to achieve optimal placement results. This paper builds on existing 2D sensor placement algorithm [4, 5] and the latest VGP spatial modelling technologies [14, 15]. Its value is found primarily in the pairing of technologies that in turn improve the existing methods of sensor placement.

The choice of the technologies used in this work are detailed and argued in the following points:

- **Preprocessing input distribution** A Masked Autoregressive Flow (MAF) is used to normalise the training dataset suffering from being non-Gaussian distributed [20] making our methodology greatly generalised as well as improving the accuracy.

- **Spatial model** A major challenge facing scalable sensor placement is overcome by deploying a Variational Gaussian Process (VGP), using a low rank approximation that is far more scalable and also addresses the question of model generalisation. In particular, this helps to tackle the limiting $O((mN)^3)$, high polynomial time complexities associated with GPs to $O((mN)\hat{N}^2)$ where $\hat{N}$ denotes the number of approximate posterior samples computed in the VGP. Using a VGP is a good trade-off between efficiency and accuracy [14, 15].

- **Sensor placement algorithm** The Mutual Information (MI) based placement algorithm [4, 5] is extended with a Markov-Chain Monte Carlo (MCMC) wrapper to fine-tune the sensor placement and tackle the time complexity $O(N^4)$ and achieve $O(klM^4)$, where k is the number of sensors, l is the number of iteration needed to optimise the position and M is such that $M << N$. The use of MCMC leads to similar placement results in a fraction of the computation time required when not using it. Error propagation through the system does exist due to this approximation, however, it is shown in the paper to be a worthwhile trade-off.

The technologies used in this paper are general and are not limited to the test case of sensor placement in indoor environments, even though their integrated implementation was designed accordingly. In fact, the core underlying technology used in the

spatial model, i.e. Variational Gaussian Process, has been successfully deployed for a multitude of other domains ranging from kriging to robotics [27].

In addition to the new pairing of technologies proposed in this paper, the novelties of this work also lie in:

- **The simulated training data** In order to achieve scalable and reduced cost deployment of sensor placement optimisation, simulation data was used for model training, replacing the expensive option of collecting large amounts measurements from installed sensors. As an example, the training dataset used in [5] consists of 52 sensors, located on a 2D plan. In this paper, the simulated training dataset is much larger, i.e. more reliable, and consist of 10,000 sensor locations distributed in 3D.
- **The increase in dimensionality** This paper increases the dimensionality of the learning problem in order to capture further correlations between hidden features as well as the output features, resulting in 3-dimensional spatial placement. The 3D placement made it possible for the model to capture more realistic and complex physical phenomena such as thermal stratification for example.
- **The fine-tuning of sensor placement** The MCMC wrapper algorithm is used to increase the overall Mutual Information captured. The base set of potential sensor location is fine-tuned to include other regions in the continuous space having higher MI. This means the selection pipeline is a more optimal set of possible placement coordinates to choose from. Additionally, the implementation is easy to be customised and makes our methodology generalisable.
- **The fully parallel and scalable implementation** The methodology presented in this paper is done through a multi-threaded parallel implementation. The computational graph based implementation, using the TensorFlow library [28], greatly speeds up the computational times.

3. The Variational Gaussian Process for optimal sensor placement (VGPosp) model

This section introduces the theoretical concepts and mathematical formulation that were developed and implemented as part of the proposed model architecture.

Let X be the solution of a dynamic system:

$$\dot{X} = F(X, t) \tag{3.1}$$

where, t denotes the time and $X = \Big[X_1, \ldots, X_m\Big]$ denotes a vector of m state variables[1] such that $X_i \in \Re^N$, $\forall i = 1, \ldots, m$. In the following, $X_t = X(t)$ denotes the solution of the dynamic system at time t and $X_{i,t} = X_i(t)$ the i-th state variable at time t.

Given a temporal sequence $X_{t_1}, \ldots, X_{t_n}$ of n solutions of the dynamical system defined in equation (3.1), with $X_{t_i} = [X_{1,t_i}, \ldots, X_{m,t_i}]$ $\forall i = 1, \ldots, n$, the Variational Gaussian Process for optimal sensor placement (VGPosp) model consists of the following main steps described in the sub-sections 3.1 to 3.3.

- Section 3.1 - **Preprocessing and preconditioning**
  - Preconditioning of input vector distributions using Masked Autoregressive Flows (MAF).
  - Convert time-series to vector value distributions at distinct spatial regions.
- Section 3.2 - **Variational Gaussian Process**
  - Sampling algorithm, input vectors spatial training.
  - Variational Gaussian Process (VGP) training.
- Section 3.3 - **Placement algorithm**
  - Selection of a set S of considered input coordinates.
  - Generate target vectors with VGP inference at coordinates.
  - Covariance matrix of values indexed by set S.
  - Greedy selection algorithm of coordinates with maximal Mutual Information (MI).
  - Markov-Chain Monte Carlo (MCMC) based fine-tuning of coordinates.

3.1. **Pre-processing and preconditioning.** The temporal sequence $X_{t_1}, \ldots, X_{t_n}$ requires a pre-processing to be suitable for the spatial model and the inference step during the placement algorithm. The pre-processing consists of the following steps, also described in Figure 1:

(1) X is first normalised to a mean value of 0 and standardised to achieve a standard deviation of each feature of 1.

[1]For example, as shown in Section 4, for fluid dynamic simulations in indoor environment, $X = [T, P, C]$, where T is the temperature, P the pressure and C the pollutant concentration.

(2) Secondly, a Masked Autoregressive Flow (MAF) is implemented to learn invertible, non-linear transformations between the non-Gaussian distributed features and the target Gaussian family of distributions.

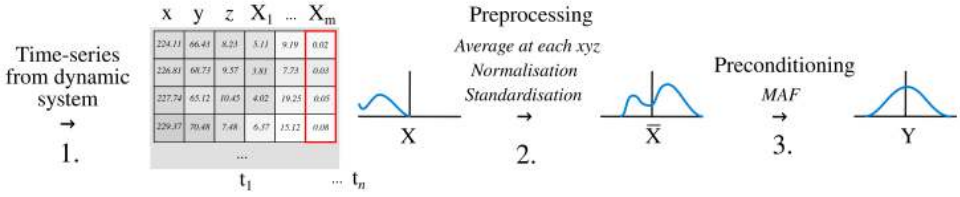


FIGURE 1. Pre-processing and pre-conditioning steps.

Firstly, the mean of the temporal sequence for each state variable X_i, $i = 1, \ldots, m$, is computed, i.e. the vector

$$\overline{X} = \left[\overline{X}_1, \ldots, \overline{X}_m\right] \tag{3.2}$$

where

$$\overline{X}_i = \frac{\sum_{j=1}^{n} X_{i,t_j}}{n}, \qquad \forall i = 1, \ldots, m \tag{3.3}$$

The vector $\overline{X}$ in equation (3.2) is used to train the Masked Autoregressive Flows (MAF) in order to make our input variables Gaussian distributed. MAF is a neural network that executes a normalising flow non-linear transformation at each neuron. MAF can be also computed as a stack of autoregressive Masked Autoencoder for Distribution Estimation (MADE) networks [20, 24] where each model uses the vector $\overline{X}$ in equation (3.2). The Autoregressive property of MAF defined from time-series analysis, predicts a future value of a variable from a linear combination of its past values. Each MADE learns the distribution of the state variables.

The MAF model can be defined as follows [22]:

$$\overline{X}_N = f_N\Big(\overline{X}_{N-1}, \overline{X}_{N-2}, \ldots, \overline{X}_1\Big) \tag{3.4}$$

where f_N has a polynomial form such that [20]:

$$\overline{X}_N = \theta_0 + \theta_1 \overline{X}_{N-1} + \theta_2 \overline{X}_{N-2} + \cdots + \theta_p \overline{X}_{N-p} \tag{3.5}$$

where θ_i are the polynomial coefficients.

Normalising flows apply a sequence of N invertible, differentiable transformation functions f_N in $p(\overline{X})$. A base distribution $p(\overline{X}')$ is specified most likely from the family of Gaussian distributions [20], where p denotes the probability distribution. The procedure begins with this initial distribution $p(\overline{X}')$ [23].

$$p(\overline{X}) = p(\overline{X}') \left| \det \frac{\partial f^{-1}}{\partial \overline{X}'} \right| \tag{3.6}$$

A chain rule can then be applied to the conditionals of a joint distribution.

$$p(\overline{X}) = \prod_{i=1}^{N} p\Big(\overline{X}_i | \overline{X}_{i-1}, \overline{X}_{i-2}, ..., \overline{X}_1\Big) = \prod_{i=1}^{N} p\Big(\overline{X}_i | \overline{X}_{<i}\Big) \tag{3.7}$$

After each non-linear transformation the distribution becomes more complex. Sampling from this transformed distribution is done via the flow of a straightforward sample from the original $p(\overline{X})$ Gaussian distribution through the non-linear transformations. The entropy of the resultant distribution is computed with the logarithm of the transformations:

$$\log p(\overline{X}_N) = \ln p(\overline{X}_0) - \sum_{i=1}^{N} \ln \left| \det \frac{\partial f_N}{\partial \overline{X}_{i-1}} \right| \tag{3.8}$$

$$p(Y) = \prod_{i=1}^{N} \left(f_N^{-1}(Y) \right) \left| \det \frac{\partial f_N^{-1}}{\partial Y} \right| \tag{3.9}$$

The vector

$$Y = \Big[Y_1, \ldots, Y_m \Big] \in \Re^{m \times N}, \quad \text{with} \quad Y_j \in \Re^N \ \forall j = 1, \ldots, m \tag{3.10}$$

is the set of normalised state variables which are input of the Variational Gaussian Process introduced in next section.

3.2. **VGP training and inference.** Given the data set $\{Y_j\}_{j=1}^m$ of $Y_j \in \Re^N$ as defined in equation (3.10), the m source-target pairs $\mathcal{D} = \{(Y_j, T)\}_{j=1}^m$, where T is a target[2] [29]. We aim to learn a function over all source-target pairs:

$$T = g(Y_j) \tag{3.11}$$

[2]In some application, $T = Y_j$ for a fixed j can be assumed.

where $g : \mathbb{R}^N \to \mathbb{R}^N$ is unknown. Let the function g decouple as $g = (g_1, ..., g_N)$, where each $g_i : \mathbb{R}^N \to \mathbb{R}$. A GP regression [12] estimates the functional form of g by placing a prior,

$$p(g) = \prod_{i=1}^{N} \mathcal{GP}\Big(g_i; 0, \Sigma_{ij}\Big), \tag{3.12}$$

where Σ_{ij} denotes a covariance evaluated over pairs of inputs $Y_i Y_j \in \mathbb{R}^N$ [12]:

$$\Sigma_{ij} = \frac{\sum_k Y_{ik} Y_{jk}^T}{N} \tag{3.13}$$

A variational Gaussian process (VGP) is a Bayesian non-parametric variational model that admits arbitrary structures to match posterior distributions. As described in the following steps, the VGP generates approximate posterior samples Z by generating latent inputs, warping them with random non-linear mappings, and using the warped inputs as parameters to a mean-field distribution. The random mappings are drawn conditional on variational parameters. The VGP specifies a generative process for posterior latent variables Z. At the first step it draws latent input $\xi \in \mathbb{R}^N : \xi \sim \mathcal{N}(0, I)$ and a non-linear mapping $g : \mathbb{R}^N \to \mathbb{R}^N$ conditioned on $\mathcal{D} : g \sim \prod_{i=1}^{N} \mathcal{GP}(0, \Sigma_{\xi\xi})|\mathcal{D}$. Then it draws approximate posterior samples $Z \in supp(p) : Z = (Z_1, ..., Z_{\hat{N}}) \sim \prod_{i=1}^{\hat{N}} q(g_i(\xi))$. Marginalising over all latent inputs and non-linear mappings, the VGP is [29]:

$$q_{VGP}(Z; \theta, \mathcal{D}) = \iint \left[\prod_{i=1}^{\hat{N}} q\Big(Z_i | g_i(\xi)\Big)\right] \left[\prod_{i=1}^{\hat{N}} \mathcal{GP}\Big(g_i; 0, \Sigma_{\xi\xi}\Big)|\mathcal{D}\right] \mathcal{N}(\xi; 0, I) df d\xi. \tag{3.14}$$

The VGP is parametrised by kernel hyperparameters θ and variational data [29, 12]. The random function interpolates the values in the variational data, which are optimised to minimise the Kullback–Leibler divergence [30]. It defines a measure between two probability density functions: $q_{VGP}(Z; \theta, \mathcal{D})$ and $q^\star(Z|Y)$, where $q^\star(Z|Y)$ is the posterior distribution [31].

$$\begin{aligned} D_{KL}\Big(q_{VGP}(Z; \theta, \mathcal{D}) || q^\star(Z|Y)\Big) &= \mathbb{E}_q \left[\log \frac{q_{VGP}(Z; \theta, \mathcal{D})}{q^\star(Z|Y)}\right] \\ &= \mathbb{E}_q \Big[\log q_{VGP}(Z; \theta, \mathcal{D}) - \log q^\star(Z|Y)\Big] \end{aligned} \tag{3.15}$$

where [19] $\mathbb{E}[f(x, \theta)] = \int_\theta f(x, \theta) d\theta$, where f denotes a distribution function. An approximating distribution is chosen from a predefined family of distributions with

parameters: θ.

$$q_\theta(Z|Y) = argmin_\theta D_{KL}\Big(q_{VGP}(Z;\theta,\mathcal{D})||q^\star(Z|Y)\Big) \tag{3.16}$$

When the KL-divergence converges to 0 (see the Theorem of the Universal approximation in [29]), the posterior is approximately the same as the learned posterior. Minimising KL-divergence is done via the objective function, defined by the Evidence Lower Bound (ELBO), parameterised by the parameters θ. The maximisation of this function is equivalent as minimising KL with a difference measured by a constant factor [23]. The objective is summed over all data-points [19]:

$$\text{ELBO}(\theta) = \sum_{i=1}^{\hat{N}} \mathbb{E}_{q_\theta(Z|X_i)}[\log q_{VGP}(Z;C,Y_i) - \log q_\theta(Z|Y_i)] \tag{3.17}$$

When applied to the marginal probability of the evidence [29, 15], the objective lower bound on the marginal likelihood can be quantified by the ELBO:

$$\text{ELBO}(\theta) = \mathbb{E}_q[\log q_{VGP}(Z;\theta,X)] - \mathbb{E}_q[\log q_\theta(z|x)] \tag{3.18}$$

Posterior is therefore the sum of the ELBO and the KL term:

$$\log q(Z) = \text{ELBO}(\theta) + D_{KL}(q_\theta(Z|Y)||q^\star(Z|Y)). \tag{3.19}$$

An approximate solution in the mean-field [32] is sought to learn parameters of the marginal likelihood [33], obtaining an approximate posterior distribution of the true posterior. The mean-field approximation of variational inference allowed for the approximate $q_\theta(Z|Y)$ distribution to be considered as a factor of $\hat{N}$ independent latent variable partitions $q_\theta(Z|Y_i)$.

$$q^\star(Z|Y) \approx q_\theta(Z|Y) = \prod_{i=1}^{\hat{N}} q_\theta(Z|Y_i) \tag{3.20}$$

then it yields $g \sim q^\star(Z|Y)$ [34].

3.3. **Placement Algorithm.** In this section,the Placement algorithm of VGPosp model is introduced. The algorithm computes optimal coordinates for sensor placement following three main steps as described in Figure 2 and detailed in Algorithm 1.

The set of coordinates $\mathcal{V}$ initially consists of N grid point: $|\mathcal{V}| = N$. Our placement algorithm is mainly based on the Mutual Information (MI) based placement algorithm [4, 5] which is extended in this paper with a MCMC wrapper to fine-tune

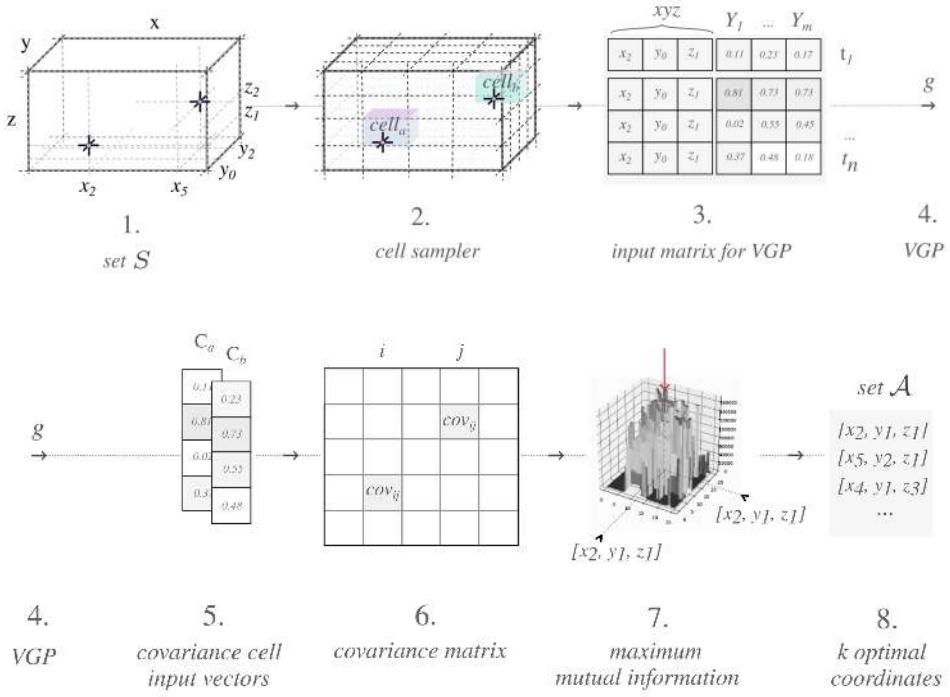


FIGURE 2. Graphical representation of the main steps of the Variational Gaussian Process for optimal sensor placement (VGPosp) model

the sensor placement and tackle the time complexity $O(N^4)$ and achieve $O(klM^4)$, where k is the number of sensors, l is the number of iteration needed to optimise the position and M is a predefined small subset of N such that $M << N$. In fact, the fist step of Algorithm 1 consists of identifying all the possible locations which constitutes a set S, $|S| = M$ (see points 1 and 2 in Figure 2). The set S is an input of Algorithm 1. Other inputs are the number k of sensors to place and the number l of maximum iterations for the optimisation process. We implement an optimisation method to identify a set $\mathcal{A}$ as the placement output from the predefined set $\mathcal{S}$ of input coordinates [4] such that $|\mathcal{A}| = k$. The second step of Algorithm 1 consists in sampling m state variables from value distributions in Y at corresponding spatial regions/cells (see point 3 in Figure 2). For each state variable in Y (see point 4 in Figure 2), the samples T are produced by the function g which is computed by a

Algorithm 1: The Variational Gaussian Process for optimal sensor placement (VGPosp) algorithm.

Input: number of sensors: k, space of coordinates: S, maximum number of iterations: l; ϵ

```
1  A* = ∅                                   ▷ initialise the set of optimal sensors locations
2  Σ = ∅                                    ▷ initialise the covariance matrix
3  δ_ȳ = 0                                  ▷ initialise the mutual information parameter
4  while ii < k do
5      while it < l do
6                                           ▷ compute the covariance matrix
7          for i ∈ S do
8              for t_k ∈ [t_0, t_n] do
9                  T_ik = g(Y_ik)           ▷ VGP function in 3.11
10             T_i = [T_i0, T_i1, ..., T_in]
11             for j ∈ S do
12                 for t_k ∈ [t_0, t_n] do
13                     T_jk = g(Y_jk)       ▷ VGP function in 3.11
14                 T_j = [T_j0, T_j1, ..., T_jn]
15             compute Σ_ij                 ▷ using equation (3.13)
16         A, δ_i* ← Algorithm 2 (Σ, S, k)  ▷ estimate the mutual information
17         if δ_i* > δ_ī or A* = ∅ then
18             A* = A
19             δ_ī = δ_i*
20         it++
21     ii++
22 return A*
```

VGP as described in section 3.2 (see points 9-14 in Algorithm 1):

$$\forall j \in S, \quad T_{jk} = g(Y_{jk}), \quad T_j = [T_{j0}, T_{j1}, \ldots, T_{jn}]$$

Then the covariance matrix of the values T_j are computed for the locations specified in S using equation (3.13) (see points 5 and 6 in Figure 2). We also define the covariance matrices related to a subset $\mathcal{A}$ such that:

$$\Sigma_{i\mathcal{A}} = \begin{bmatrix} \Sigma_{i1} & 0 & 0 & \ldots \\ 0 & \Sigma i2 & 0 & \ldots \\ \vdots & \ddots & \ddots & \vdots \\ \ldots & 0 & 0 & \Sigma_{ik} \end{bmatrix} \tag{3.21}$$

where Σ_{ij} is defined in equation (3.13) and $k = |\mathcal{A}|$. $\Sigma_{i\mathcal{A}}$ is used in the final step of Algorithm 1 which maximise the mutual information (see points 7 and 8 in Figure 2), as described in Algorithm 2, where H is the conditional entropy function defined as (see [5]):

$$H(i|\mathcal{A}) = \frac{1}{2} log \Sigma^2_{i|\mathcal{A}} + \frac{1}{2}(log(2\pi) + 1) \tag{3.22}$$

and where δ_i denotes the mutual information parameter [5]:

$$\delta_i = \frac{\Sigma^2_{ii} - \Sigma_{iA}\Sigma^{-1}_{AA}\Sigma_{Ai}}{\Sigma^2_{ii} - \Sigma_{i\bar{A}}\Sigma^{-1}_{\bar{A}\bar{A}}\Sigma_{\bar{A}i}} \tag{3.23}$$

Algorithm 2: Maximise Mutual Information (MI) using lazy evaluations.

Input: Covariance matrix Σ_{ij}, number of sensors k, set of coordinates S

$\mathcal{A} \leftarrow \emptyset$
foreach $i \in \mathcal{S}$ **do**
  $\delta_i \leftarrow +\infty$
for *j=1 to k* **do**
  foreach $i \in \bar{\mathcal{A}} = \mathcal{S} \setminus \mathcal{A}$ **do**
    $current_i \leftarrow false$
  while $current_i == true$ **do**
    $i^* \leftarrow argmax_{i \in S \setminus \mathcal{A}}\ \delta_i$
    if $current_{i^*} == true$ **then**
      break
    Compute $\Sigma_{i\mathcal{A}}$ ▷ using equation (3.21) for i and $\mathcal{A}$
    Compute $\Sigma_{i\bar{\mathcal{A}}}$ ▷ using equation (3.21) for i and $\bar{\mathcal{A}}$
    $\delta_{i^*} \leftarrow H(i|\mathcal{A}) - H(i|\bar{\mathcal{A}})$ ▷ with H defined in (3.22)
    $current_{i^*} \leftarrow true$
  $\mathcal{A} \leftarrow \mathcal{A} \cup i^\star$
return $\mathcal{A}$

We developed a MCMC based wrapper that works with a smaller grid of points $\mathcal{S}$, then identifies the vertices of the grid $\mathcal{A}$ that are potential frontiers to be explored further. These optimal coordinates in $\mathcal{S}$ are thus adjusted. If the adjustment has improved the overall mutual information we keep the modification. Using this technique we iteratively adjust $\mathcal{S}$ until the mutual information value converges.

3.4. **Implementation.** A code implementing Algorithm 1 and Algorithm 2 using TensorFlow.1.4 [28] is available at https://github.com/roxarcucci/VGPosp.git. Instructions for running the tests and algorithms are described in the file README.md. The algorithms were initially implemented in Python and reimplemented later in TensorFlow in order to further improve the efficiency of the run-time through multi-threaded, parallelised execution and automatic scalability to more computational cores. Our implemented model is represented in the form of a computational data-flow graph that is instantiated once a session object is defined. The built-in TensorFlow compiler identifies all dependencies within our algorithms and assigns multi-threaded computational tasks to our resources. TensorBoard creates a visual representation of the nodes and connections, it enables the developer to debug connectivity errors. The Scalars tool that allows the tracking of any metric of interest during model training or optimisation was also used.

4. Results and Discussions

In this section, two test cases are used to discuss our Variation Gaussian Process for optimal sensor placement (VGPosp):

- The first test case, named the sine model, is a simplified two-dimensional model of a sine function to test the efficiency, accuracy and precision of VGP compared to GP. The comparison between GP and VGP is done only for the sine model as the complexity of GP is too high to be trained for a real test case.
- The second test case is a real three-dimensional test case considering a room within the Clarence Centre building located in Elephant and Castle, London, UK. In this test case, the predictive model is the Computational Fluid Dynamics (CFD) software Fluidity. Optimal sensors location is proposed. The benefit of using sensors optimally located is proved by showing how the predictive model error can be more efficiently reduced by using Data Assimilation technology.

4.1. **Test case 1: Sine model.** This section aims to compare the efficiency, the accuracy and the precision of GP and VGP. The test sine function used is defined in equation(4.1).

$$f(x,y) = \sin\left(\frac{2}{3}\pi x\right) + \sin\left(\frac{2}{3}\pi y\right) \tag{4.1}$$

In order to prove that VGP is more efficient than a GP approach, the training time as a function of the number of training points for both method are shown in Figure 3.

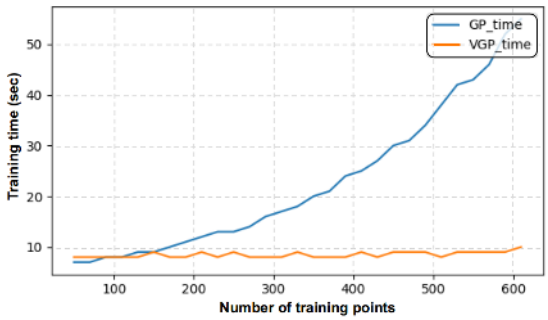


FIGURE 3. Training time as a function of number of training points when using a Gaussian Process (GP) or a Variation Gaussian Process (VGP).

Up to 150 number of training points, the GP and the VGP method both take a little bit less then 10 sec to be trained. However, when using more training points, i.e. more than 150 number of points, the training time of the GP increases drastically, while the VGP training time stays constant, around 10 sec, independently of the number of points. For example, when considering 600 training points, the training time is divided by 5.5 when using the VGP approach.

The accuracy e and the root mean squared error (also called precision), $RMSE$ of the model $\mathcal{M}$ are:

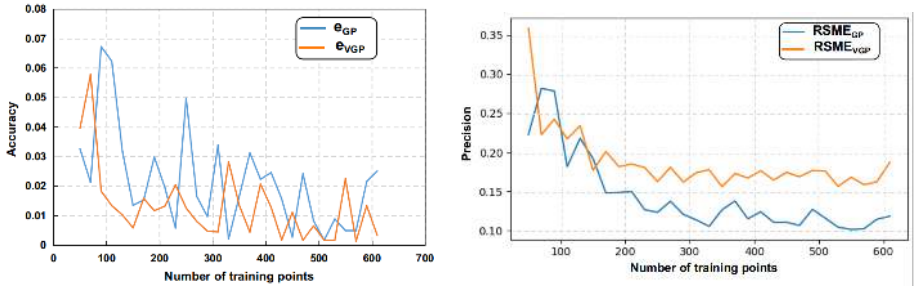
$$e_{\mathcal{M}}(N) = \sum_{i=1}^{N} |f_{True}(x_i, y_i) - f_{\mathcal{M}}(x_i, y_i)| \tag{4.2}$$

where N is the number of training points, and

$$RMSE_{\mathcal{M}}(N) = \sqrt{\frac{\|F_{\mathcal{M}} - F_{True}\|_{L^2}}{\|F_{True}\|_{L^2}}} \tag{4.3}$$

where $F_{\mathcal{M}}$ and F_{True} denote the vectors $F_{\mathcal{M}} = [f_{\mathcal{M}}(x_1, y_1), \ldots, f_{\mathcal{M}}(x_N, y_N)]$ and $F_{\mathcal{M}} = [f_{True}(x_1, y_1), \ldots, f_{\mathcal{M}}(x_N, y_N)]$, the $True$ model denotes the function in equation (4.1) and the model $\mathcal{M}$ stands for GP or VGP.

The accuracy (equation (4.2)) and the precision (equation (4.3)) of the two models as a function of the number of training points are shown in Figure 4. From Figure 4a, when using less than 150 training points, the two models highlight the worst accuracy, i.e. the highest values. When using more than 150 number of training points, it can be seen than the accuracy of GP is lower than 0.03, while the VGP accuracy is lower



(A) (B)

FIGURE 4. (A) Accuracy e and (B) Precision $RMSE$ as a function of number of training points when using a Gaussian Process (GP) and a Variation Gaussian Process (VGP).

than 0.02. Globally, the VGP method is slightly more accurate than GP even if the accuracy can be considered as the same order of magnitude. However, looking at Figure 4b, the GP model is more precise than the VGP. For both model, the RMSE is relatively high when using less than 200 number of training points. When using more training points, the RMSE reaches a plateau with values for the GP and VGP model of about 0.12 and 0.17, respectively.

Overall, it has been shown that the VGP method is a good trade-off between the efficiency, the accuracy and the precision and will then be used as such an assumed tool in the second test case.

4.2. **Test case 2: Real test case.**

4.2.1. *Predictive model: Computational Fluid Dynamics simulation using Fluidity software.* The simulated data used to train our VGPosp is obtained using Fluidity, a parallel open-source CFD software (http://fluidityproject.github.io/). It uses finite elements to solve the following incompressible three dimensional Navier-Stokes equations, continuity equation (4.4) and momentum equation (4.5), on unstructured grids [35]:

$$\nabla.\overline{u} = 0 \tag{4.4}$$

$$\frac{\partial \overline{u}}{\partial t} + \overline{u}.\nabla\overline{u} = -\frac{1}{\rho}\nabla\overline{p} + \nabla.\Big[\left(\nu + \nu_\tau\right)\nabla\overline{u}\Big] \tag{4.5}$$

where $\overline{u}$ is the resolved velocity (m/s), $\overline{p}$ is the resolved pressure (Pa), ρ is the fluid density (kg/m^3), ν is the kinematic viscosity (m^2/s) and ν_τ is the anisotropic eddy

viscosity (m^2/s).
Turbulence is resolved using Large Eddy Simulation (LES), where the eddies smaller than a scale Δ are parametrised using a subgrid-scale module, while the larger eddies are fully resolved. The subgrid-scale model in Fluidity is based on the Smagorinsky model [36, 37].

The transport of a scalar field C (i.e, a passive tracer or pollutant concentration) in kg/m^3 is expressed using the advection-diffusion equation (4.6):

$$\frac{\partial C}{\partial t} + \nabla.(\mathbf{u}C) = \nabla.\left(\overline{\overline{\kappa_C}}\nabla C\right) + F \tag{4.6}$$

where $\mathbf{u}$ is the velocity vector (m/s), $\overline{\overline{\kappa_C}}$ is the diffusivity tensor of the pollutant in an excess of air (m^2/s) and F represents the source terms ($kg/m^3/s$).
The temperature field T (Kelvin) is expressed using equation (4.7):

$$\frac{\partial T}{\partial t} + \nabla.(\mathbf{u}T) = \nabla.\left(\overline{\overline{\kappa_T}}\nabla T\right) + \frac{Q}{\rho c_p} \tag{4.7}$$

where $\mathbf{u}$ is the velocity vector (m/s), $\overline{\overline{\kappa_T}}$ is the thermal diffusivity tensor (m^2/s), Q represents thermal source terms (W/m^3), ρ is the fluid density (kg/m^3) and c_p is the fluid specific heat capacity (J/kg/K). The behaviour of the atmospheric boundary layer in Fluidity is represented using a turbulent inlet velocity based on a synthetic eddy method [38, 39]. Fluidity uses mesh adaptivity where the mesh can be dynamically refined, during the simulation, in areas of physical significance to the user [40].

4.2.2. *Test case set up description.* The test case considered in this paper is a room within the Clarence Centre building located at London South Bank University (LSBU) near Elephant and Castle in London, UK (Figure 5). The test room has three windows depicting in blue in Figure 5. This test site was used to conduct a one-day field study in January 2018 with the MAGIC project (http://www.magic-air.uk/, [41]) during which 7 sensors were monitoring the indoor temperature and CO_2 concentration. The CFD simulation performed in this paper aims to replicate a cross ventilation scenario, where the test room windows on both sides of the building were opened.

As shown in Figure 6, the computational domain considered to do the numerical simulations includes the entire Clarence building and the test room, as well as the immediate building upwind in order to replicate the local flow conditions near the windows. The mesh is defined such that the resolution is increased in the room (setting the grid edge length to 0.1 m) and particularly at the openings (grid edge length set to 0.02 m). It progressively decreases in the overall domain to reach an

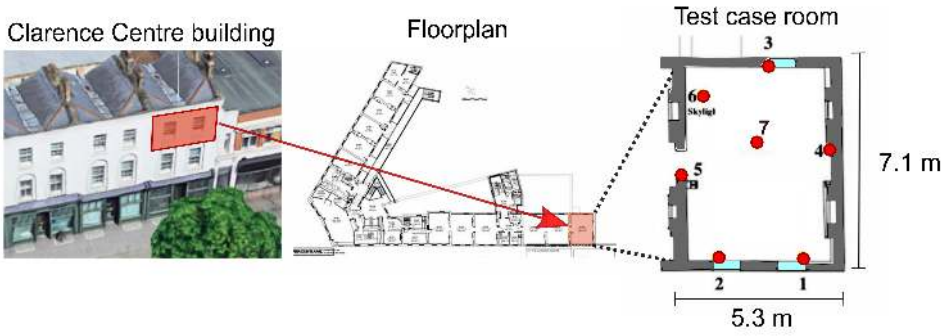


FIGURE 5. The test case room is located in Clarence Centre building in London, UK. The red dots denotes the location of sensors during a field experiment and blue rectangles shows the location of the three windows.

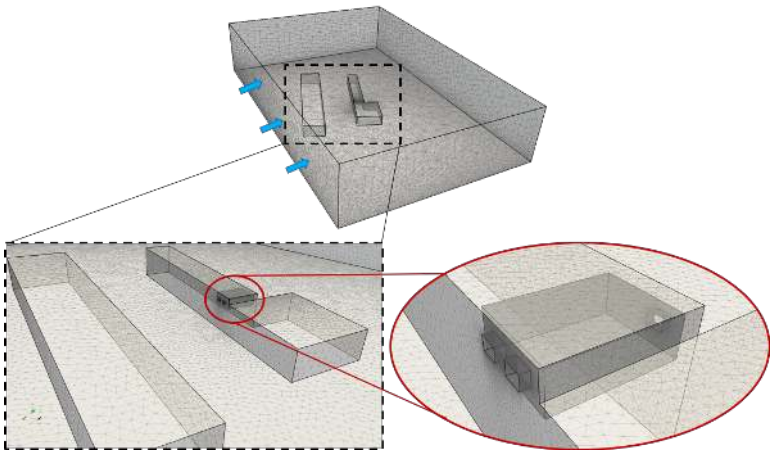
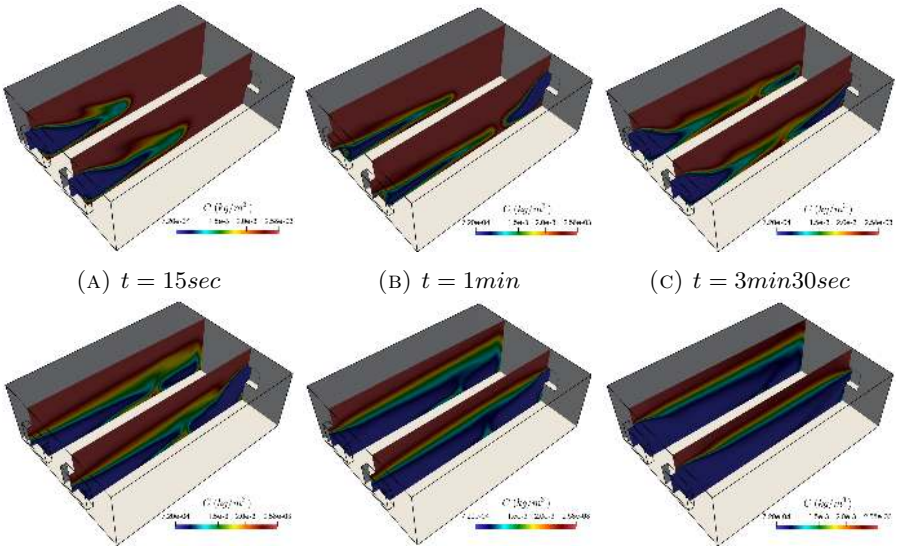


FIGURE 6. Computational domain and surface mesh of the area of interest showing the Clarence Centre and the upwind building as well as the test case room. The blue arrows denote the wind direction.

edge length of 10 m away from the room as shown in Figure 6, which gives an overall number of 285,700 cells, i.e. grid points, in the mesh. The total number of nodes within the room is about 1.5×10^5.

The boundary conditions are set to replicate the experimental conditions. A log-law turbulent inlet velocity is imposed upwind, corresponding to a wind direction of 201°. It is parametrised with an incoming wind velocity of 2.58 m/s at 28.5 m. No slip



(A) $t = 15sec$ (B) $t = 1min$ (C) $t = 3min30sec$

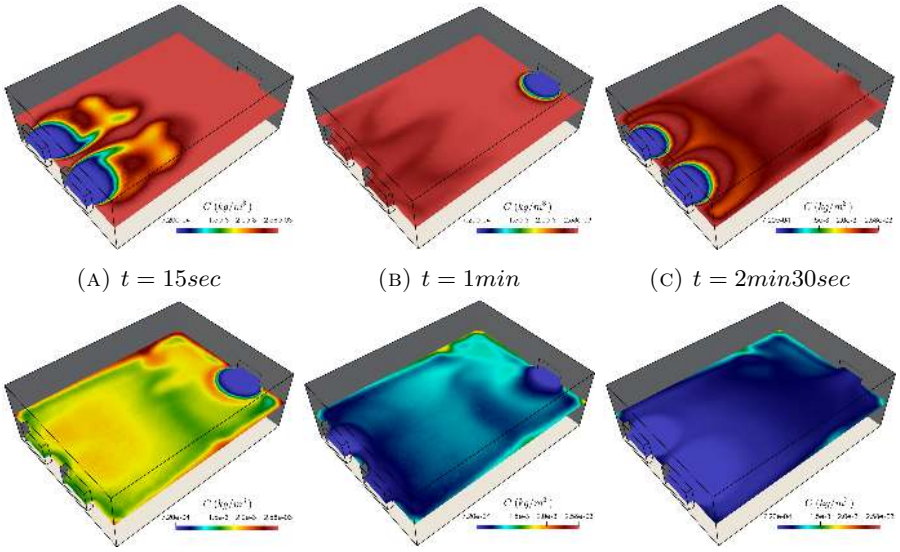
(D) $t = 5min$ (E) $t = 10min$ (F) $t = 15min$

FIGURE 7. Concentration field of CO_2 on two vertical slices in the room at different time. The scale is between $7.2 \times 10^{-4} kg/m^3$ (blue colour) and $2.58 \times 10^{-3} kg/m^3$ (red colour).

boundary conditions are imposed at the bottom of the domain and on the walls of the test room. The initial temperature is set to 19.5 °C inside the room and 9.1 °C outside. Before opening the windows, based on sensors data, the average CO_2 concentration in the room is set to $2.58{\times}10^{-3}$ kg/m^3 (1420 ppm) while $7.2{\times}10^{-4}$ kg/m^3 (400 ppm), is prescribed outside as a background pollution level.

The simulation was run in parallel on 20 CPU and for an overall simulation time of 15 min, leading to about 3500 timesteps. In this paper, the target variables is the concentration C of CO_2 within the room. As an example, the evolution of the concentration field on different planes in the room within the room at different times are shown in Figure 7 and Figure 8. At the beginning of the simulation, the outdoor air enters the room gradually and concentration stratification starts to occur after 2 minutes 30 seconds. It can be seen that the concentration in the room starts to reach a steady state after 5 minutes and does not change anymore after 15 minutes of simulations.

4.2.3. *Results.* The training set consists in 10,000 points, exceeding the amount used in previous work by a factor of 150 [4, 5]. The quantity and 3D spatial positioning of



(A) $t = 15sec$ (B) $t = 1min$ (C) $t = 2min30sec$

(D) $t = 5min$ (E) $t = 10min$ (F) $t = 15min$

FIGURE 8. Concentration field of CO_2 on an horizontal plane in the room at different time. The scale is between $7.2 \times 10^{-4} kg/m^3$ (blue colour) and $2.58 \times 10^{-3} kg/m^3$ (red colour).

the training set was sufficient to capture the phenomenology of the indoor environment. The CFD results used to train the VGP are taken between 2 min 30 sec and 5 min, period during which the stratification of the concentration is established. The solution of this dynamical system X in (3.1) includes the physical variables $[P, T, C]$, where P is the pressure, T the temperature and C, the physical variables target, the CO_2 concentration.

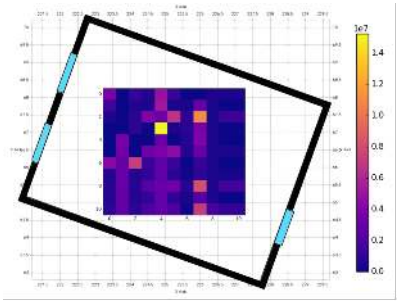
The simulated training data had features with non-Gaussian likelihoods, causing potential problems for spatial learning with Gaussian Processes. While the use of a Variational Gaussian Process helps overcome this issue, we further generalised our training data through the use of a Masked Autoregressive Flow model that transforms the likelihoods of the input features to the family of Gaussian-distributions.

The results of the optimal sensor placement are discussed in the following. Firstly, the scalability issues related to dense grid initialisations of set S are addressed. Secondly, the introduction of the MCMC based fine-tuning algorithm is motivated. Finally, the optimal sensor placement solutions provided by our proposed VGPosp model are presented.

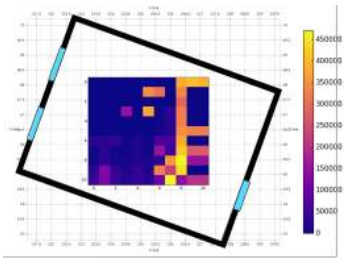
The placement Algorithm 2 is executed on a predefined area of interest, in which a density parameter specifies the grid initialisation that defines set S. Figure 9 demonstrates the internal state, i.e. mutual information parameter, of the placement algorithm before making a selection. For each coordinate δ_i is computed (Algorithm 2) and the most optimal coordinate is selected into the final selection set $\mathcal{A}$. For example, in Figure 9a, the first sensor will be chosen to be located where δ_i is the highest, i.e. yellow colour part. From Figure 9, it can be seen that the scale of δ_i is not the same in each sub-figures and becomes narrower. Indeed, the contributions made by earlier selections are higher, explaining why the δ_i quantities decrease after more sensors are added to set S.

The running-times shown in Figure 10 are descriptive of the exponential computational cost that is incurred from selecting a larger input set of S. However, it cannot be expected from the selection Algorithm 1 to find the optimal locations in continuous space when defining only distant, discrete points. In order to reduce the time complexity of discovering optimal coordinates in continuous space, we proposed an MCMC-based fine-tuning method of set S. The impact of this procedure is demonstrated by Figure 11, where the percentage increase in the optimisation criterion, mutual information parameter can be multiple orders of magnitude larger than that of the uniform grid instantiation, which may miss more optimal regions. For example in the case of a $7 \times 7 \times 1$ grid instantiation, the fine-tuning Algorithm 1 can achieve a 3 order of magnitude improvement in the overall mutual information, associated with our final selection set A. This improvement is achieved after 15 grid optimisation attempts. In Figure 12, a $6 \times 6 \times 4$ grid is considered associated with 40 grid optimisation attempts for each selection. The optimal selection of 7 sensors are plotted with crosses as well as all modified grid positions that had achieved improved mutual information parameter values. Their colour corresponds with the mutual information parameter that the coordinate had achieved. The final optimal coordinates of the room are depicted in Figure 13.

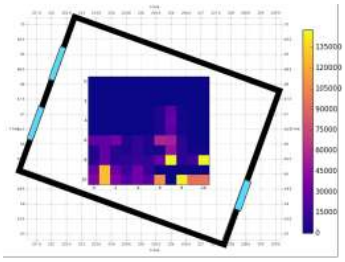
Data Assimilation technology is coupled with the predictive model Fluidity. Data Assimilation uses observed data from sensors to improve and correct the numerical results from the simulation. 7 sensors are assimilated in the predictive model. The DA algorithm and methodology used was previously successfully coupled with Fluidity and is presented in details in [2, 42]. The accuracy of the DA results are evaluating using the mean squared error: $MSE(C) = \frac{\|C - C^{v,n}\|_{L^2}}{\|C^{v,n}\|_{L^2}}$, where C is either C^n the Fluidity concentration at time step n or C^{DA} the corrected concentration using DA and $C^{v,n}$ is the control variable, i.e. the true observed data. The MSE is computed using the 7 optimal sensor locations shown in Figure 13 and compared



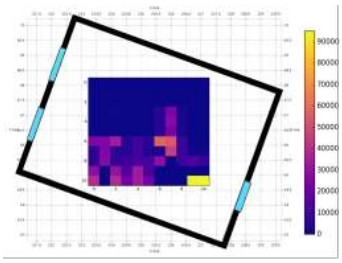
(A) Sensor 1



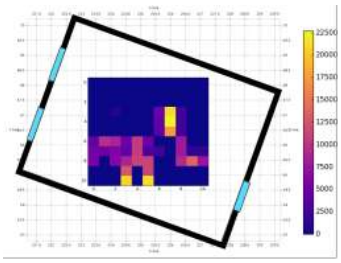
(B) Sensor 2



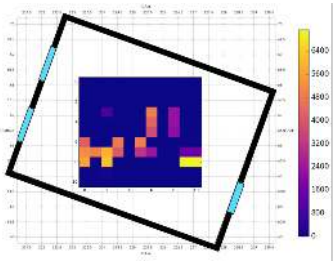
(C) Sensor 3



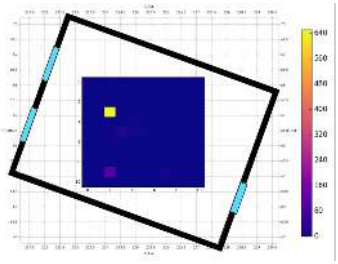
(D) Sensor 4



(E) Sensor 5



(F) Sensor 6



(G) Sensor 7

FIGURE 9. Mutual Information parameter δ_i on M grid points ($11 \times 11 \times 1$) for $k = 7$ sensors. The colours show the value of δ_i and the scale is different on each sub-figures. Yellow and violet colours denote high and low δ_i values respectively.

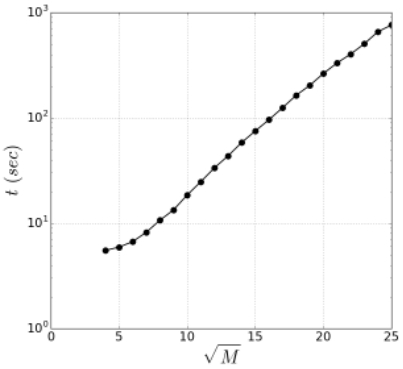


FIGURE 10. Execution time t of Algorithm 1 as a function of the number of initial grid points M. The y-axis is logarithmic.

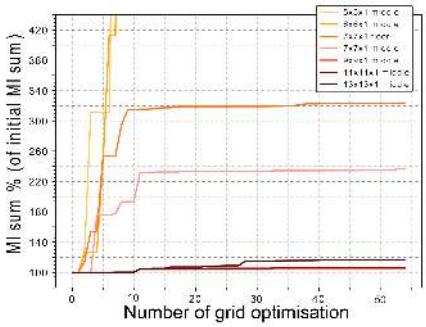


FIGURE 11. Percentage increase in mutual information parameter after the fine-tuning Algorithm 1.

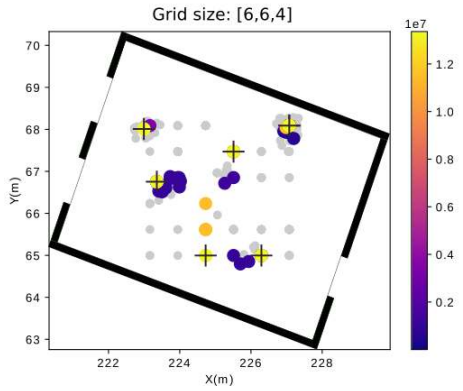


FIGURE 12. Fine-tuning of grid-points and final selection coordinates. Colour represents mutual information parameter of placement. The colour of the point shows the value of the Mutual Information. The crosses depict the final optimal sensor locations.

with the MSE obtained using 7 sensors located randomly. 2000 random sensors positioning were performed. Assimilating the seven optimally positioned sensors, the error of the predictive model, i.e. Fluidity, is reduced by up to three order of magnitude: $MSE(C^n) = 0.17$ and $MSE(C^{DA}) = 0.0005$. Moreover, this error is

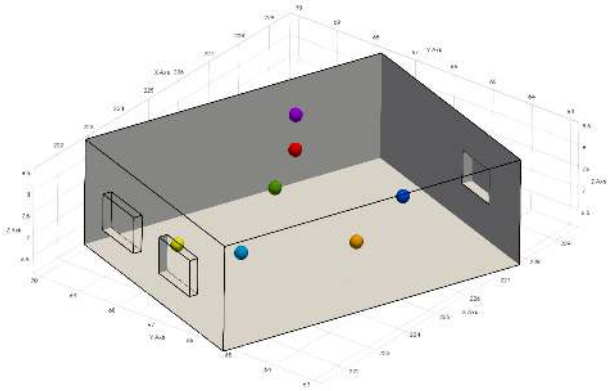


FIGURE 13. Optimal final location of seven sensors in the room of Clarence Centre obtained using the Variational Gaussian Process optimal sensor placement VGPosp in Algorithm 1.

up to two order of magnitude lower than the ones computed using random sensors placement. In one of the worst random case scenario: $MSE(C^{DA}) = 0.023$.

5. Conclusion

This work described a novel pipeline for sensor placement, incorporating a Masked Autoregressive Flows (MAF) [20] for preconditioning, a Variational Gaussian Process (VGP) [14, 15] spatial model and a mutual information-based placement algorithm [5]. In this paper, the alterations to the existing placement pipelines are significant as they introduce multiple layers of approximations in order to reduce time complexities. More specifically, VGP was introduced for the sensor placement pipeline to tackle the $O(N^3)$ complexity associated with traditional GP for spatial modelling. Increased model generalisation was achieved with MAF that learn non-linear invertible transformations between complicated transformations and the more flexible Normal distribution. All models and algorithms were implemented as a TensorFlow computational graph to further reduce run-times as opportunities for parallel computation are automatically recognised by the graph compiler. Furthermore, this work proposed and developed two extension algorithms. One focused on incorporating information and sampling environmental features from the time-series for computing mutual information. Secondly, a wrapper algorithm was built to iteratively subsampling and globally optimise the instantiated set of base grid-coordinates, leading to a three-fold increase in mutual information associated with the final selection. The combination of these two algorithms achieved stability and a global improvement in the selection coordinates.

Acknowledgments

This work is supported by the EPSRC Grand Challenge grant Managing Air for Green Inner Cities (MAGIC) EP/N010221/1 and by the EPSRC Centre for Mathematics of Precision Healthcare EP/N0145291/1

References

[1] Kelly Frank J. and Fussell Julia C. Improving indoor air quality, health and performance within environments where people live, travel, learn and work. *Atmospheric Environment*, 200:90–109, 2019.

[2] Rossella Arcucci, Laetitia Mottet, Christopher Pain, and Yi-Ke Guo. Optimal reduced space for variational data assimilation. *Journal of Computational Physics*, 379:51–69, 2019.

[3] L D'Amore, R Arcucci, L Marcellino, and A Murli. A parallel three-dimensional variational data assimilation scheme. In *AIP Conference Proceedings*, volume 1389, pages 1829–1831. AIP, 2011.

[4] Carlos Guestrin, Andreas Krause, and Ajit Paul Singh. Near-optimal sensor placements in gaussian processes. In *Proceedings of the 22nd international conference on Machine learning*, pages 265–272. ACM, 2005.

[5] Andreas Krause, Ajit Singh, and Carlos Guestrin. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, 9(Feb):235–284, 2008.

[6] Rossella Arcucci, Luisa D'Amore, Jenny Pistoia, Ralf Toumi, and Almerico Murli. On the variational data assimilation problem solving and sensitivity analysis. *Journal of Computational Physics*, 335:311–326, 2017.

[7] Héctor González-Banos. A randomized art-gallery algorithm for sensor placement. In *Proceedings of the seventeenth annual symposium on Computational geometry*, pages 232–240. ACM, 2001.

[8] Kumar Abhishek, MP Singh, Saswata Ghosh, and Abhishek Anand. Weather forecasting model using artificial neural network. *Procedia Technology*, 4:311–318, 2012.

[9] Dan Cornford, Ian T. Nabney, and Christopher KI Williams. Adding constrained discontinuities to gaussian process models of wind fields. In *Advances in Neural Information Processing Systems*, pages 861–867, 1999.

[10] Cheng-Chun Lin and Liangzhu Leon Wang. Forecasting simulations of indoor environment using data assimilation via an ensemble kalman filter. *Building and Environment*, 64:169–176, 2013.

[11] David JC MacKay. Introduction to gaussian processes. *NATO ASI Series F Computer and Systems Sciences*, 168:133–166, 1998.

[12] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer School on Machine Learning*, pages 63–71. Springer, 2003.

[13] Joaquin Quiñonero-Candela and Carl Edward Rasmussen. A unifying view of sparse approximate gaussian process regression. *Journal of Machine Learning Research*, 6(Dec):1939–1959, 2005.

[14] Michalis K. Titsias. Variational model selection for sparse gaussian process regression. *Report, University of Manchester, UK*, 2009.

[15] Michalis Titsias. Variational learning of inducing variables in sparse gaussian processes. In *Artificial Intelligence and Statistics*, pages 567–574, 2009.

[16] John Hagan, AR Gillis, and Janet Chan. Explaining official delinquency: A spatial study of class, conflict and control. *The sociological quarterly*, 19(3):386–398, 1978.

[17] James Hensman, Nicolo Fusi, and Neil D. Lawrence. Gaussian processes for big data. *arXiv preprint arXiv:1309.6835*, 2013.

[18] Haitao Liu, Yew-Soon Ong, Xiaobo Shen, and Jianfei Cai. When gaussian process meets big data: A review of scalable gps. *arXiv preprint arXiv:1807.01065*, 2018.
[19] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
[20] George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, pages 2338–2347, 2017.
[21] Carl Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.
[22] Hadley Wickham. *ggplot2: elegant graphics for data analysis*. Springer, 2016.
[23] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.
[24] Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. Made: Masked autoencoder for distribution estimation. In *International Conference on Machine Learning*, pages 881–889, 2015.
[25] Noel Cressie. Statistics for spatial data. *Terra Nova*, 4(5):613–617, 1992.
[26] Satish Tadepalliy Naren Ramakrishnany, Chris Bailey-Kellogg and Varun N. Pandey. Gaussianprocessesfor active dataminingof spatial aggregates. In *SIAM Data Mining*, pages 427–438, 2005.
[27] Botond Bócsi, Philipp Hennig, Lehel Csató, and Jan Peters. Learning tracking control with forward models. In *2012 IEEE International Conference on Robotics and Automation*, pages 259–264. IEEE, 2012.
[28] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.
[29] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017.
[30] Solomon Kullback and Richard A. Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.
[31] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.
[32] Viet Hung Tran. Copula variational bayes inference via information geometry. *arXiv preprint arXiv:1803.10998*, 2018.
[33] Matthew James Beal et al. *Variational algorithms for approximate Bayesian inference*. University of London, 2003.
[34] Dustin Tran, Rajesh Ranganath, and David M Blei. The variational gaussian process. *arXiv preprint arXiv:1511.06499*, 2015.
[35] AMCG. Fluidity manual (Version 4.1), 2015.
[36] J. Smagorinsky. General Circulation Experiments With the Primitive Equations. *Monthly Weather Review*, 91(3):99–164, 1963.
[37] T. Bentham. *Microscale modelling of air flow and pollutant dispersion in the urban environment*. PhD thesis, Imperial College London, 2004.
[38] D. Pavlidis, G.J. Gorman, J.L.M.A. Gomes, C. Pain, and H. ApSimon. Synthetic-Eddy Method for Urban Atmospheric Flow Modelling. *Boundary-Layer Meteorology*, 136:285–299, 2010.
[39] N. Jarrin, S. Benhamadouche, D. Laurence, and R. Prosser. A synthetic-eddy-method for generating inflow conditions for large-eddy simulations. *International Journal of Heat and Fluid Flow*, 27(4), 2006.
[40] Christopher Pain, A. P. Umpleby, C. R.E. de Oliveira, and A. J.H. Goddard. Tetrahedral mesh optimisation and adaptivity for steady-state and transient finite element calculations. *Computer Methods in Applied Mechanics and Engineering*, 190(29-30):3771–3796, 2001.

[41] Jiyun Song, S. Fan, William Lin, L. Mottet, H. Woodward, M. Davies Wykes, R. Arcucci, D. Xiao, J-E Debay, H. ApSimon, et al. Natural ventilation in cities: the implications of fluid mechanics. *Building Research & Information*, 46(8):809–828, 2018.

[42] E. Aristodemou, R. Arcucci, L. Mottet, A. Robins, C. Pain, and Y.-K. Guo. Enhancing CFD-LES air pollution prediction accuracy using data assimilation. *Building and Environment*, 165:106383, 2019.

Accronyms

CFD: Computational Fluid Dynamics

DA: Data Assimilation

ELBO: Evidence Lower BOund

GP: Gaussian Process

IAQ: Indoor Air Quality

KL: Kullback-Leibler

LES: Large Eddy Simulation

LSBU: London South Bank University

MADE: Masked Autoencoder for Distribution Estimation

MAF: Masked Autoregressive Flows

MCMC: Markov-Chain Monte Carlo

MI: Mutual Information

SGP: Sparse Variational Process

VGP: Variational Gaussian Process

VGPosp: Variational Gaussian Process optimal sensor placement

Authors addresses: *Gabor Tajnafoi*, Data Science Institute, Dept. of Computing, Imperial College London, UK, e-mail: `gabor.tajnafoi18@imperial.ac.uk`. *Rossella Arcucci*, Data Science Institute, Dept. of Computing, Imperial College London, UK, e-mail: `r.arcucci@imperial.ac.uk`. *Laetitia Mottet*, Dept. of Earth Science & Engineering, Imperial College London, UK, e-mail: `l.mottet@imperial.ac.uk`. *Carolanne Vouriot*, Dept. of Civil Engineering, Imperial College London, UK, e-mail: `carolanne.vouriot12@imperial.ac.uk`. *Miguel Molina-Solana*, Dept. of Computer Science and AI, Universidad de Granada, Spain, e-mail: `miguelmolina@ugr.es`. *Christopher Pain*, Dept. of Earth Science & Engineering, Imperial College London, UK, e-mail: `c.pain@imperial.ac.uk`. *Yi-Ke Guo*,

Data Science Institute, Dept. of Computing, Imperial College London, UK, e-mail: y.guo@imperial.ac.uk.