A Decentralised On-Line Coordination Mechanism for Monitoring Spatial Phenomena with Mobile Sensors

Ruben Stranders, Alex Rogers and Nicholas R. Jennings
Electronics and Computer Science, University of Southampton, Southampton, SO17 1BJ, UK.
{rs06r, acr, nrj}@ecs.soton.ac.uk

ABSTRACT

In this paper, we introduce the first *on-line, decentralised* coordination mechanism for monitoring spatial phenomena with multiple mobile sensors that model the environment using Gaussian processes (GPs). These sensors have their application domain in highly dynamic scenarios, where strict time constraints prohibit path planning in advance, or where the characteristics of the spatial phenomena are unknown a priori. Our coordination mechanism enables the sensors to learn the environmental parameters on-line from their collected observations (and represents them using the GP), and to coordinate their movements to maximise their collective information gain in a decentralised way. We evaluate our algorithm using real-world sensor data collected at the Berkeley Intel Research lab, and demonstrate that a team of 5 mobile sensors equipped with our algorithm monitors the environment with a 51% reduction of Root Mean Squared error compared to a network of fixed sensors that have been deployed using an off-line optimisation algorithm that has complete information of the characteristics of the spatial phenomena.

1. INTRODUCTION

In disaster response, and many other applications besides, the availability of timely and accurate information is of vital importance. Thus, the use of multiple mobile sensors for information gathering in crisis situations has generated considerable interest[1]. These mobile sensors could be autonomous ground robots or unmanned aerial vehicles. In either case, while patrolling through the disaster area, such as a smoke-filled building, these sensors usually need to keep track of changing environmental conditions, such as the temperature, and the concentration of smoke and potentially toxic chemicals. The key challenges in so doing are twofold. First, the sensors cannot cover the entire environment at all times, so the spatial and temporal dynamics of the monitored phenomena need to be identified in order to predict measurements in the rest of the environment. Second, the sensors need to coordinate their movements to collect the most informative measurements needed to identify these dynamics.

Recent work has attempted to solve these challenges using Gaussian processes (GPs). A GP is a principled Bayesian method for inference about functions [8], and has been shown to be an effective tool for capturing the dynamics of spatial phenomena [1]. Using GPs in this fashion, researchers have presented algorithms to calculate informative placements of fixed sensors [3], and for pre-planning informative paths for multiple mobile sensors [9].

Whilst these approaches present a significant contribution to their field, they are less suitable in our domain. First, these algorithms are geared towards solving a one-shot optimisation problem in an off-line phase. In rapidly changing environments, however, mobile sensors need to be able to continuously adapt to changing circumstances. Consequently, this is an ongoing optimisation problem that requires an on-line solution. Second, they assume the availability of a set of previously collected measurements from which the dynamics of the spatial phenomena can be learnt. This assumption does not hold in disaster response scenarios, because in most cases no sensor data is available beforehand (because it is usual a unique situation that has not been seen before), and they are inherently uncertain. Third, these algorithms do not model the temporal dynamics of spatial phenomena; they consider how a phenomenon varies in space, but not in time. For example, the temperature in an environment may be fairly similar at different locations in the environment at any given moment, but change rapidly over time. For mobile sensors in a rapidly changing environment, such dynamics are very important. Specifically, it allows them to determine whether the uncertainty in the environmental conditions at an unvisited location has grown too large, and consequently whether that location needs to be revisited. Finally, these algorithms are centralised. In hostile environments, this is undesirable, because it creates a single point of failure, thereby increasing the vulnerability of the information stream.

To address these shortcomings, we present the first on-line, decentralised coordination mechanism for teams of mobile sensors in which path planning and learning are simultaneously performed on-line. To this end, we represent each sensor as an autonomous learning agent. These agents are capable of making measurements, exchanging observations with other agents, and modelling both the spatial and temporal characteristics of the phenomena without prior knowledge. By exchanging their observations, the agents share the same world view. This allows the agents to avoid each other, and choose their movements to maximise collective information gain. More specifically, the agents are able to coordinate because they share the same world view through the exchange of observations. As a result, each agent is able to determine the information content at every location in the environment at every moment in time. For example, an observation made by an agent reduces the amount of information directly around it. Each individual agent will recognise this and move away from others towards more informative parts of the environment. Moreover, because every agent controls its own movements using information it possesses locally, there is no

[1]For example, one of the missions of both the ALADDIN project (`http://www.aladdinproject.org`) and the Center for Robot Assisted Search & Rescue (`http://crasar.csee.usf.edu`) is to use autonomous robots for information gathering in disaster response scenarios.

central point of control, and the coordination mechanism is decentralised.

In more detail, in this paper we contribute to the state of the art in the following ways:

- We devise an algorithm that allows mobile sensors to coordinate on-line through the exchange of observations. From these observations, each agent learns the *hyperparameters* of its GP. This GP not only models the spatial, but also the temporal dynamics of the phenomena being monitored. These models are then used by the agents to move autonomously through the environment in search of informative observations.
- We evaluate our approach using real-world sensor data from the Intel Berkeley Research lab. The results of this simulation show that use of our coordination mechanism in a team of 5 mobile sensors results in a 51% reduction of root mean squared error compared to a network of fixed sensors, or equivalent performance to a fixed sensor network of 15 nodes (in both cases, the fixed sensors are placed optimally assuming prior knowledge of hyperparameters using an off-line optimization routine).

The remainder of the paper is organised as follows. First, we present a formalisation of the problem domain. Then, we describe our decentralised coordination mechanism and empirically evaluate it.

2. THE MONITORING PROBLEM

Our informal discussion of the problem of monitoring spatial phenomena above gives rise to the requirements that our mobile sensors need to satisfy. Specifically, we want them to autonomously collect observations in the environment in order to accurately monitor it. To this end, the sensors should:

- Process noisy observations and identify temporal and spatial trends in the environment.
- Coordinate to collect observations that maximise the information gain of the collective.

More formally, we denote an environment by a triple $\mathcal{E} = (\mathcal{S}, \mathcal{G}, \mathcal{P})$, where

- $\mathcal{S} = \{s_i | i = 1 \dots N\}$ is the set of N mobile sensors;
- The graph $\mathcal{G} = (V, E)$ defines legal moves: the set V contains all locations the sensors can visit, and edge-set $E \subseteq V \times V$ contains all permissible moves between locations in V;
- $\mathcal{P}$ is a spatial phenomena that is monitored by the sensors in $\mathcal{S}$. Here, we model phenomenon $\mathcal{P}$ as some real valued function of time and location: $\mathcal{P} : \mathcal{V} \times T \rightarrow \mathbb{R}$.

The sensors' locations at time t are denoted by the N-tuple $L_t = \left(l_t^1, \dots, l_t^N\right)$, where $l_t^i \in \mathcal{V}$. At given times t, the sensors take measurements $O_t = \left(o_t^1, \dots, o_t^N\right)$ at locations L_t by sampling from $\mathcal{P}$: $o_t^i \leftarrow \mathcal{P}(l_t^i, t)$.

Given this model, the sensors' challenge is to monitor $\mathcal{P}$ at all locations $\mathcal{V}$ at time t. Since the number of sensors N is generally much smaller than $|V|$ (the number of locations that are monitored), the sensors need to not only take measurements at locations S_t, but also *predict* the value of $\mathcal{P}$ at time t for every location V, based on observations $O_{t' < t}$ made earlier. We denote these predictions at time t by $P_t = \{p_t^v | v \in V\}$. Suppose the actual measurements made at those locations are $A_t = \{a_t^v | v \in V\}$, then the challenge faced by the sensors is to coordinate their movements to minimise the root mean squared error (RMSE) of the predictions for all timesteps t:

$$RMSE(P_t, A_t) = \sqrt{\frac{\sum_{v \in V} (a_t^v - p_t^v)^2}{|V|}} \tag{1}$$

3. THE DECENTRALISED COORDINATION ALGORITHM

In this section, we describe our decentralised coordination algorithm. Before doing so, however, we give a brief overview of GPs, show how the hyperparameters of a GP can be learnt from observations, and show how to quantify informative placements of sensors.

3.1 Gaussian Process Basics

As mentioned earlier, GPs have been shown to provide an effective tool for modelling spatial phenomena. One of their key features is the ability to predict a measurement at any location given a set of previously collected observations (either in space or in time), with an explicit representation of the certainty of that prediction. More formally, given a set of n observations $\mathcal{A} = \{(\mathbf{x}_i, y_i) | i = 1 \dots n\}$, where $\mathbf{x}_i$ denotes a D dimensional input vector, and y denotes the target output, the GP can predict the value y_* for any input vector $\mathbf{x}_*$. This value is normally distributed: $p(y_* | \mathcal{A}) = \mathcal{N}(y_*; \mu, \sigma^2)$, where μ and σ^2 are obtained using the following equations.

$$\mu = \mu_{y_*} + \Sigma_{y_* \mathcal{A}} \Sigma_{\mathcal{A}\mathcal{A}}^{-1} (y_{\mathcal{A}} - \mu_{\mathcal{A}}) \tag{2}$$

$$\sigma^2 = \mathcal{K}(y_*, y_*) - \Sigma_{y_* \mathcal{A}} \Sigma_{\mathcal{A}\mathcal{A}}^{-1} \Sigma_{\mathcal{A} y_*} \tag{3}$$

Here, $\mathcal{K}(\cdot, \cdot)$ is referred to as the covariance function, and the elements of the covariance matrix Σ_{XY} are computed by evaluating $\mathcal{K}(\cdot, \cdot)$ for all pairs of elements from sets X and Y. In this context, the covariance function determines how observations are correlated, and is critically dependent on the characteristics of the spatial phenomena of interest. For most spatial phenomena, the correlation between observations is inversely proportional to their separation in the input space (in space, time or both). Moreover, these phenomena tend to be non-smooth functions of time and location. To model these kinds of functions, the Matérn class of covariance functions has been shown to be suitable [6] and we will, therefore, adopt it in this work:

$$\mathcal{K}(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \left(1 + \sqrt{3} r\right) \exp\left(-\sqrt{3} r\right) \tag{4}$$

where σ_f^2 is the signal variance, and r is defined as:

$$r \equiv \sqrt{(\mathbf{x} - \mathbf{x}') \mathbf{P}^{-1} (\mathbf{x} - \mathbf{x}')} \tag{5}$$

Here, $\mathbf{P}$ is a diagonal matrix with entries $l_1^2,\ldots,l_D^2$, that scale the dimensions of the input vector $\mathbf{x}$ independently. Depending on the type of dimension, these entries are more commonly referred to as *lengthscales* or *timescales*. The more gradually the modelled phenomenon varies over an input dimension, the longer lengthscale for that dimension, and vice versa[2]. The individual scales allow us to model processes that are strongly correlated along one input dimension, while weakly correlated along another. For example, a spatial phenomena that varies slowly over space, but very quickly over time has a long lengthscale, but a short timescale[3].

As formalised earlier, the phenomenon $\mathcal{P}$ is a function of time and space. Therefore, we use three input dimensions $\mathbf{x} = (x, y, t)$: two spatial (corresponding to the x and y dimensions of the environment), and one temporal (corresponding to the time t) to model the correlations in $\mathcal{P}$. Consequently, the matrix $\mathbf{P}$ contains three non-zero entries: $\mathbf{P} = diag(l_s^2, l_s^2, l_t^2)$, where l_s is the lengthscale, and l_t is the timescale. By adding a timescale, we explicitly introduce the temporal aspect of the spatial phenomenon into the GP, in contrast to the placement algorithms for fixed sensors (e.g. [3]).

Finally, one of the requirements discussed above is the ability of a sensor to process noisy measurements. In order to model this noise, we add an extra term $\sigma_n \delta_{\mathbf{x}\mathbf{x}'}$ to Equation 4, where σ_n is the noise variance, and $\delta_{\mathbf{x}\mathbf{x}'}$ is the Kronecker delta which is one iff $\mathbf{x} = \mathbf{x}'$. To see why this models noise, note that σ_n will appear on the diagonal of the covariance matrix, which is the variance of the measurement.

3.2 Marginalising Hyperparameters

The Matérn covariance function in Equation 4, extended with the capability of modelling noisy observations, has a number of free variables (σ_f, σ_n, l_s, l_t). Since these variables determine the distribution of weights of an underlying parametric model [8, Section 2.1], they are usually referred to as *hyperparameters*. In most cases, the values of these hyperparameters are not known *a priori*, but have to be inferred from the set of observations $O_{t'<t}$. In practical terms, this involves the marginalisation of the hyperparameters, a process captured in the following integral:

$$p\left(y|\mathcal{A}\right) = \frac{\int p(y|\mathcal{A},\phi)p(\mathcal{A}|\phi)p(\phi)d\phi}{\int p(\mathcal{A}|\phi)p(\phi)d\phi} \tag{6}$$

In our case, the parameter space ϕ is the Cartesian product of the parameter spaces of σ_f, σ_n, l_s, l_t: $\mathbb{R}^4$. Needless to say, in general, this space grows very large as the number of parameters increases. Furthermore, the non-trivial dependence between the likelihood of the parameters $p(\mathcal{A}|\phi)$ and the prediction $p(y|\mathcal{A},\phi)$ on ϕ, makes this integral non-analytic. However, Osborne *et al.* ([6]) show how sophisticated quadrature can be used to approximate it. In general, quadrature involves evaluating both $p(\mathcal{A}|\phi)$ and $p(y|\mathcal{A},\phi)$, for multiple samples of ϕ, which is a computationally intensive operation[4]. Moreover, the size of the parameter space ϕ precludes exhaustive sampling from these functions. Therefore, *Bayesian Monte Carlo* (BMC) [7] is used to reduce the number of samples. Unlike many frequentist approaches, BMC uses as much information about the sample functions as possible. So, instead of taking into account only the sample values, BMC also uses the sample locations and the smoothness of the integrand. As a result, BMC needs significantly fewer samples to produce an accurate approximation of the left hand side of Equation 6. The result of applying BMC is a weighted sum of GPs:

$$p(y|\mathcal{A}) \approx \sum_{\phi} w_\phi p(y|\mathcal{A},\phi) \tag{7}$$

Note that $p(y|\mathcal{A},\phi)$ denotes a GP fitted to data $\mathcal{A}$ for hyperparameter sample ϕ, and w_ϕ denotes the weight assigned to the GP with hyperparameters ϕ. To put it differently, the prediction of y is a weighted sum of predictions from GPs with the hyperparameters from the sample set. Space prohibits a full treatment of BMC, and how the weights w_ϕ are obtained. For further details, refer to [6] and [7]. It suffices to say that by using algorithms for re-using the major part of the computations when new data points are collected, BMC is a very suitable and efficient algorithm for learning the hyperparameters, while at the same time providing a principled way for regression and prediction[5].

3.3 Quantifying Informative Placements

Before we turn to the key contribution of this paper, we first need to quantify how informative a sensor placement is. Two measures have been proposed in earlier work. The first is the entropy criterion[6] [1]:

$$L_t^* = \arg\max_{L_t} H(L_t) \tag{8}$$

where L_t is obtained from L_{t-1} through legal moves of the mobile sensors (i.e. moving sensor s_i from l_{t-1}^i to l_t^i is possible iff $(l_{t-1}^i, l_t^i) \in E$). Using this criterion, the sensors will position themselves at locations with high entropy at every time step t[7].

[2]Note that for $l_i = 1$ for $1 \leq i \leq D$, the numerator of the exponent becomes the square of the Euclidean distance between the two vectors, in which case the resulting GP exhibits the same characteristics over all input dimensions.

[3]This choice is not critical to our work, but has been found to be suitable during our experimental evaluation. Different phenomena may warrant other types of covariance functions, without the need for changing the operation of our proposed coordination mechanism.

[4]Calculating $p(y|\mathcal{A},\phi)$ alone involves fitting a GP with parameters ϕ on observations $\mathcal{A}$ using Equations 2 and 3.

[5]Simpler alternatives for BMC exist for estimating hyperparameters ϕ, such as Marginal Likelihood (ML) [8, Section 5.4]. However, our experiments showed that ML too often ends up in local maxima, resulting in very poor predictions.

[6]With slight abuse of notation, we will denote the location and the random variable representing the measurement at that location with the same symbol.

[7]Guestrin, Krause, & Singh ([3]) show that this criterion tends to place fixed sensors along the border of the environment, where they are maximally uncertain about each others' measurements. Consequently, these sensors waste a large part of their sensor range. As an alternative, they propose the mutual information (MI) criterion, which leads to a more central placement of the sensors: $L_t^* = \arg\max_{L_t} I(V \setminus L_t; L_t)$, where mutual information $I(X;Y)$ measures the reduction in uncertainty of random variable X, given the value of random variable Y. Using MI, sensors tend to collectively position themselves at those locations that reduce the uncertainty at locations $V \setminus L_t$ as far as possible. However, calculating MI is a computationally more involved process than calculating the entropy, because it requires calculating the result-

Regrettably, optimising entropy is an NP-hard problem [4]. Moreover, in the applications mentioned earlier, the observations needed to learn the hyperparameters are collected on-line, and are not available beforehand. These two considerations have led to the use of a *greedy on-line* policy [5]. The policy is *greedy*, because at each timestep, individual sensors move to the location with the highest entropy, without considering subsequent moves. This is computationally less demanding than calculating L_t^* in Equation 8. Furthermore, the policy is *on-line*, because the entropy is constantly being updated based on newly acquired observations made by the sensor itself and the other sensors. Now, we can formalise the greedy on-line entropy criterion for each individual sensor as follows:

$$l_{t+1}^* = \underset{l_{t+1} \in adj(l_t)}{\arg\max} \; H(l_{t+1} | O_{t' < t} = \mathbf{o_{t' < t}}) \tag{9}$$

where l_{t+1} is an adjacent location to the sensor's current location l_t in graph $\mathcal{G}$, and $\mathbf{o}_{t' < t}$ is the realisation of the set of variables $O_{t' < t}$, that is, all observations made before the current time step.

This leaves open the question of how entropy can be evaluated in a GP. For a random variable X that is normally distributed with variance σ^2, the entropy can be evaluated analytically:

$$H(X) = \log(\sigma\sqrt{2\pi e}) \tag{10}$$

However, the approximation of $p(y|\mathcal{A})$ using BMC in Equation 7 is not a normal distribution, but a mixture of Gaussians. No method of analytically evaluating the entropy of such a mixture is known to us. Fortunately, our experiments show that the distribution of the weights w_i becomes very peaked after only a limited number of observations. Consequently, the result effectively approximates a normal distribution and allows us to use the variance of the probability distribution function (pdf) from Equation 7[8] in Equation 10.

3.4 The Coordination Algorithm

Our algorithm is outlined in pseudo code in Algorithm 1. At the beginning of each time step, the sensors take measurements (line 1), and send these to their peers (line 2). Their own measurements, combined with those received from the other sensors in line 3, are used to update the GP model (line 4). Next, each sensor predicts measurements at every location in $\mathcal{G}$ for which it is closest (line 6). Finally, in line 8, each sensor moves to the neighbouring location with the highest entropy according to Equation 9.

This algorithm has two important emergent properties. The first is the patrolling behavior of the individual agents. Using the greedy on-line entropy policy, sensors will tend to move towards locations with high entropy. Recall that the covariance function we use not only takes into account the spatial correlations in the environment, but also the temporal correlations. As a result, observations made in the past will have increasingly less relevance for predicting the current state of the environment. The entropy at locations that remain unvisited will therefore increase automatically, and those locations consequently become increasingly attractive to visit again.

ing entropy in the entire environment for every possible movement. Moreover, preliminary experiments with mobile sensors showed no significant improvements in performance when using MI instead of entropy. So, we focus on the entropy in the remainder of this paper.

[8]Details on calculating the variance of this pdf can be found in [6].

Algorithm 1 The coordination algorithm instantiated for sensor s_i at time t.

1: Take measurement $o_t^i \leftarrow \mathcal{P}(t, l_t^i)$
2: Send measurement o_t^i and current position l_t^i to sensors $S \setminus s_i$
3: Receive measurements $O_t^{S \setminus s_i}$ and positions $L_t^{S \setminus s_i}$ from other sensors
4: Update GP using newly acquired measurements
5: **for all** $\{v \in \mathcal{V} | \; |l_t^i - v| \leq |l_t^j - v| \, \forall j \neq i\}$ **do**
6: Calculate prediction p_i^v
7: **end for**
8: Move to the adjacent vertex l_{t+1}^* that maximises the entropy: $l_{t+1}^* = \underset{l_{t+1} \in adj(l_t)}{\arg\max} \; H(l_{t+1} | O_{t' < t} = \mathbf{o_{t' < t}})$

This effectively incentivises the sensors to be in a constant state of hill climbing in the direction of the steepest entropy gradient[9]. So, in contrast to algorithms that calculate informative placements for fixed networks, it is crucial for the mobile sensors to model the temporal dynamics of the spatial phenomena, because the problem they face is not a one-off optimisation problem; the sensors need to determine the next informative placement given newly acquired observations, while the relevance of older observations declines over time. Consequently, they will have to keep patrolling the environment by revisiting previously visited locations.

The second emergent property is the coordination between the agents. The exchange of observations in step 2 of the algorithm allows the sensors to share the same world view. It enables them to calculate the entropy for every possible move, based on their neighbours' locations and observations. Sensors will therefore tend to avoid each other, and spread out, because moving in the direction of another sensor will generally decrease the entropy. As a result, the sensors are capable of implicitly coordinating their actions through the exchange of simple observations, and need not also exchange their plans or intentions.

Figure 1 shows a snapshot of the algorithm in action. It shows the path of a single sensor, and the variance with which the sensor is able to predict measurements throughout the environment. As can be seen, the sensor is moving in the direction of higher variance (and thus entropy). Note how the variance behind the sensor increases along the travelled path. This is illustrates the effect of modelling the phenomenon's time dynamics: the older the measurement at a certain location, the less useful it is to predict the current measurement. For a more dynamic view of the operation of the algorithm, see our videos at `http://www.youtube.com/mobilesensors`.

4. EXPERIMENTAL SETUP

To empirically evaluate our approach, we simulated teams of 5 sensors using a dataset from the Intel Berkeley Research lab. This dataset has been extensively used in related work to evaluate approaches in sensor networks [3]. It contains temperature, humidity and light intensity measurements, and has been collected from 54 fixed wireless sensors between 28 February and 5 April 2004. Figure 2 shows the layout of the lab, and the location of the sensors.

We compared our approach with five benchmark policies using the

[9]This is similar to *information surfing* as proposed in [2], where mobile sensors for target tracking move in the direction of the steepest mutual-information gradient.

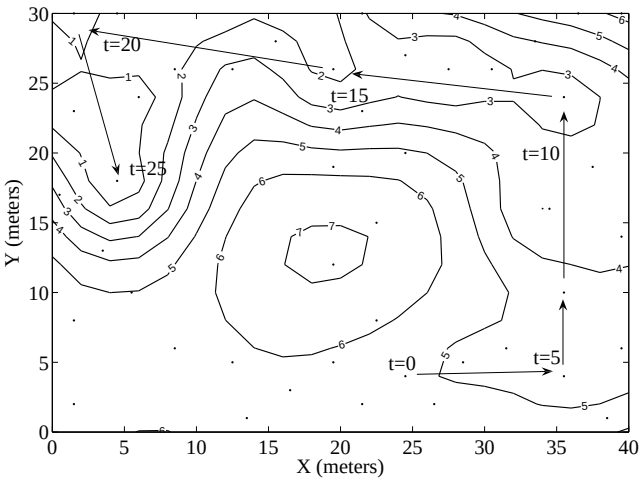

Figure 1: The world view of a single sensor moving through the Intel Berkeley Research lab at timestep $t = 25$. The path of the sensor is indicated by arrows. Superimposed is a contour plot of the predictive variance with which the sensor can predict measurements throughout the environment (the numbers on the contour lines indicate the variance). The lower the predictive variance, the better measurements at uninstrumented locations can be predicted. These predictions are made using the sensor's GP, based on measurements collected along the sensor's path. Times are in minutes.

dataset. Using the formal model introduced earlier, the graph $\mathcal{G}$ is populated with vertices V that correspond to the 54 sensor locations, and the edges E that restrict movement between any pair of locations $v_i, v_j \in V$ that are no further than 8 meters apart. At every 5 minute interval, each deployed sensor s_i makes a temperature reading o_t^i by querying the Berkeley dataset for the sensor's present location l_t^i and the current simulation time t. After the sensors have updated their GP models (Algorithm 1, step 4), the sensors make temperature predictions P_t for all of the 54 sensor locations V (step 6). The RMSE is subsequently using the actual readings A_t obtained by the 54 sensors. Finally, the policies that use mobile sensors decide where to move next.

Now, the six policies can be seen as points in three-dimensional space: the first dimension specifies the movement: mobile (M), jumping (J), and fixed (F). The second dimension describes learning: knowing (K) the hyperparameters in advance[10], or having to learn (L) them. The last dimension specifies the type of policy: greedy (G) or random (R). In case of greedy, it is also specified what is greedily maximised: entropy (e) or mutual information (mi).

MLGe Our algorithm as detailed in the previous section: mobile sensors that are capable of learning the values of the hyperparameters using BMC, and employ a greedy on-line movement policy for entropy maximization.

MLR The same mobile sensors as MLGe, except that they move randomly. This policy was included to determine the effect of the greedy entropy policy.

[10]Using an off-line learning algorithm, we determined the values of the hyperparameters of the temperature in the lab.

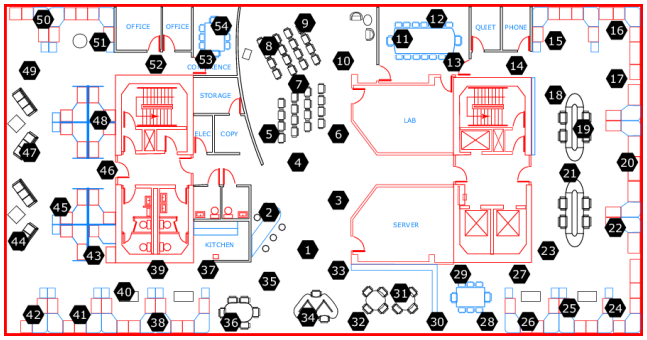

Figure 2: Sensor deployment at the Intel Berkeley Research lab. In our simulation, the sensors can move between the 54 real sensor-locations every 5 minutes to a location within 8 meters of their previous position. The lab itself measures 30 by 40 meters. (Copied from `http://db.csail.mit.edu/labdata/labdata.html`).

MKGe The same sensors as MLGe, except that they have prior knowledge of the all hyperparameters. This policy evaluates the effect of having to learn the hyperparameters.

JKGe The same as MKGe, except that these sensors can instantaneously jump to a desired location without visiting intermediate locations. This policy acts as an upper bound for achievable performance.

FKGmi Fixed sensors that are placed using a greedy mutual information maximisation algorithm that has prior knowledge of all the hyperparameters [3].

FKGe Fixed sensors that are placed using a greedy entropy maximisation algorithm that has prior knowledge of the hyperparameters.

Given the characteristics of these policies, we can formulate the following experimental hypothesis.

HYPOTHESIS 1. *The prediction accuracy of our algorithm MLGe will be higher than that of both fixed sensor networks, as well as that of the random policy.*

HYPOTHESIS 2. *Our algorithm will closely approximate the performance of benchmarks MKGe and JKGe that possess prior knowledge of all the hyperparameters, that our algorithm has to learn on-line.*

5. RESULTS

Figure 3 shows the accuracy of the predictions in terms of RMSE of a single run through the dataset, averaged over the days on which the measurements were made. It clearly shows that our mobile sensors outperform both fixed placements, and the randomly moving sensors, confirming Hypothesis 1. Furthermore, the fact that the randomly moving sensors perform worse clearly shows that the greedy entropy policy is indeed effective. Finally, the prediction accuracy of our mobile sensors is comparable to those sensors that can instantaneously jump to their desired location, and the mobile sensors that have prior knowledge of the hyperparameters (i.e.

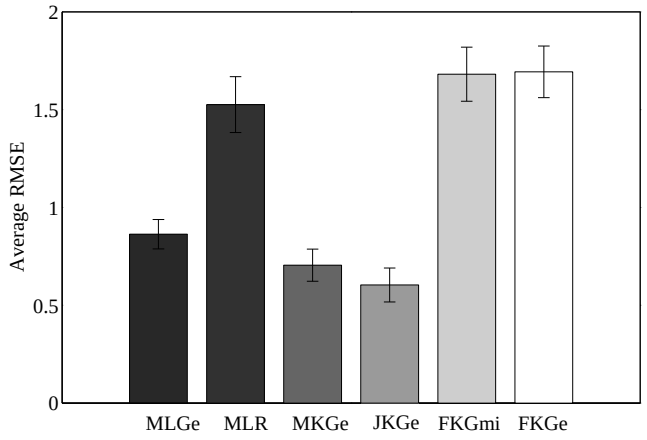

Figure 3: Average RMSE for the different types of sensor policies. Each simulation was performed with 5 sensors. The error bars indicate the standard error of the mean.

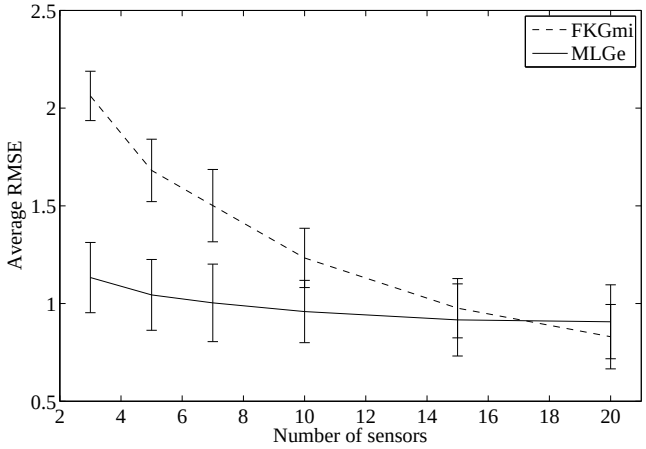

Figure 4: Graph showing the effect of varying the number of sensors in the environment. Around 15 fixed sensors are needed to attain a similar performance to the 5 mobile sensors. Additionally, the graph shows that adding additional moving sensors to the environment does not bring about a significant increase in prediction quality. The error bars indicate the standard error of the mean.

the two policies used to put an upper bound on achievable performance), confirming Hypothesis 2.

Figure 4 shows the effect of varying the number of sensors in the environment. The mobile sensors start off with a significantly higher performance than the fixed sensor network; around 15 fixed sensors with prior knowledge of the hyperparameters are needed to attain the same performance as a team of 5 mobile sensors. However, when the number of sensors increases to around 15 (30% of the number of sensors in the Berkeley lab), the two solutions become equivalent. At this point, increasing the number of fixed sensors has a greater effect on the performance than introducing additional mobile sensors. This is caused by the fact that as the number of mobile sensors increases, the freedom of movement of the sensors is reduced, thereby making it more difficult for them to reach locations of high entropy. As a result, the performance of the mobile sensors increases more slowly. We believe that this can be prevented by introducing more explicit coordination between the sensors, and we will consider this in future work.

6. CONCLUSIONS

In this paper, we introduced the first decentralised coordination mechanism for monitoring spatial phenomena with mobile sensors. These sensors can be applied in scenarios where a fixed sensor network is not available or where a sensor network needs to be rolled out quickly. Potential applications include dynamic environments, such as disaster response and surveillance, but also more routine tasks such as agricultural monitoring and weather prediction.

In particular, using a benchmark dataset, we demonstrated that not only does our algorithm monitor the phenomena better than a network of fixed sensors, but also that it does so with a reduced number of sensors. Moreover, in contrast to the fixed sensor placement algorithms, our algorithm is capable of learning the characteristics of the environment while monitoring it, without the need for an off-line learning phase. Furthermore, we showed how the sensors can coordinate their movements to collect informative measurements through greedy on-line entropy maximisation.

There are several interesting directions for future work. Firstly, we intend improve the communication protocol. For example, bandwidth can be saved by propagating only informative observations through the network, or by limiting communication to the exchange of hyperparameter weights. Also, the sensors might need to remain in communication range with each other and with the human operators that use their information. Restricting their motion to keep them connected is one of the problems that we need to address. Secondly, we would like to investigate the effect of more sophisticated coordination mechanisms. Currently, the sensors implicitly coordinate with each other through the exchange of observations, combined with zero-lookahead maximisation of the entropy. This can lead to slight complications when the number of sensors increases, as they tend to reduce each others' freedom of motion. Therefore, introducing negotiation between the sensors about planning longer paths might allow them to more effectively monitor complex environments.

7. REFERENCES

[1] N. Cressie. *Statistics for Spatial Data*. Wiley-Interscience, 1993.

[2] B. Grocholsky, A. Makarenko, T. Kaupp, and H. F. Durrant-Whyte. Scalable control of decentralised sensor platforms. In *Proceedings of the second International Workshop on Information Processing in Sensor Network (IPSN 2003)*, pages 96–112. Springer Berlin / Heidelberg, 2003.

[3] C. Guestrin, A. Krause, and A. P. Singh. Near-optimal sensor placements in gaussian processes. In *ICML '05: Proceedings of the 22nd International Conference on Machine Learning*, pages 265–272, New York, NY, USA, 2005. ACM Press.

[4] C.-W. Ko, J. Lee, and M. Queyranne. An exact algorithm for maximum entropy sampling. *Operations Research*, 43(4):684–691, 1995.

[5] A. Krause and C. Guestrin. Nonmyopic active learning of gaussian processes: an exploration-exploitation approach. In *ICML '07: Proceedings of the 24th International Conference on Machine Learning*, pages 449–456. ACM Press, New York, NY, USA, 2007.

[6] M. A. Osborne, A. Rogers, S. D. Ramchurn, S. J. Roberts, and

N. R. Jennings. Towards real-time information processing of sensor network data using computationally efficient multi-output gaussian processes. In *IPSN '08: Proceedings of the seventh International Conference on Information Processing in Sensor Networks (in press)*, 2008.

[7] C. E. Rasmussen and Z. Ghahramani. Bayesian monte carlo. In S. T. Suzanna Becker and K. Obermayer, editors, *Advances in Neural Information Processing Systems*, volume 15, pages 489–496. MIT Press, 2003.

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

[9] A. Singh, A. Krause, C. Guestrin, W. J. Kaiser, and M. A. Batalin. Efficient planning of informative paths for multiple robots. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2204–2211, 2007.