IJASS
International Journal of
Aeronautical and Space Science

Learning the Covariance Dynamics of a Large-Scale Environment for Informative Path Planning of Unmanned Aerial Vehicle Sensors

Sooho Park*

Jump Trading, Chicago, IL 60654, USA

Han-Lim Choi**

Division of Aerospace Engineering, KAIST, Daejeon 305-701, Korea

Nicholas Roy* and Jonathan P. How*****

Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA 01239, USA

Abstract

This work addresses problems regarding trajectory planning for unmanned aerial vehicle sensors. Such sensors are used for taking measurements of large nonlinear systems. The sensor investigations presented here entails methods for improving estimations and predictions of large nonlinear systems. Thoroughly understanding the global system state typically requires probabilistic state estimation. Thus, in order to meet this requirement, the goal is to find trajectories such that the measurements along each trajectory minimize the expected error of the predicted state of the system. The considerable nonlinearity of the dynamics governing these systems necessitates the use of computationally costly Monte-Carlo estimation techniques, which are needed to update the state distribution over time. This computational burden renders planning to be infeasible since the search process must calculate the covariance of the posterior state estimate for each candidate path. To resolve this challenge, this work proposes to replace the computationally intensive numerical prediction process with an approximate covariance dynamics model learned using a nonlinear time-series regression. The use of autoregressive time-series featuring a regularized least squares algorithm facilitates the learning of accurate and efficient parametric models. The learned covariance dynamics are demonstrated to outperform other approximation strategies, such as linearization and partial ensemble propagation, when used for trajectory optimization, in terms of accuracy and speed, with examples of simplified weather forecasting.

Key words: Informative planning, Unmanned aerial vehicles, Sensor networks, Learning

1. Introduction

Many natural phenomena experience rapid change. As a result, regular measurements must be taken in order to determine the current state of a natural system. Natural systems include weather, vegetation, animal herds, and coral reef health. Without frequent measurements, the difference between the estimated state of these dynamic systems and the true state can grow arbitrarily large. Sensor networks have been successfully applied and utilized in many real-world domains in order to automate the measurement process. However, but these domains possess characteristics that may limit how the sensors are deployed. Complete sensor coverage of any real-world domain is impractical because of excessive cost, time and effort. For example, it is considerably easier to deploy permanent weather sensors on land than over open ocean areas. Given finite resources, the amount of sensor information, and therefore the quality of the state estimate, is typically unevenly distributed across

(cc) * Trader
** Assistant Professor, Corresponding author
    E-mail: hanlimc@kaist.ac.kr   Tel: +82-42-350-3727   Fax: +82-42-350-3710
*** Associate Professor
**** Richard Cockburn Mclaurin Professor

the system.

The fast moving dynamics of natural domains present two challenges. Firstly, the state dynamics are strongly coupled, meaning that part of the system can often have far-reaching influence on the rest of the system. Secondly, the state dynamics are also chaotic; in other words, very small changes in the current state can lead to very large changes in the future. As a result, if some parts of the system are not sufficiently known and accurately described, the accuracy of the estimated state of the whole system is significantly reduced, as is our ability to produce predictions about the future state.

Additional measurements may improve the accuracy of state predictions. However, given finite resources, only a subset of the state can be measured with additional sensors. For mobile sensors, care must be given to choosing the locations in the system that will maximally reduce the expected error of future predictions. In particular, dynamic system prediction frequently uses a probabilistic form of filtering in which a probability distribution over the state is maintained. If the state is normally distributed with a mean and covariance, the sequence of measurements that minimizes some norm on the covariance is usually preferred.

The path selection problem (or "informative path planning" problem) may be straightforwardly solved if the posterior covariance can be computed efficiently subsequent to taking a sequence of the measurements (Berliner et al., 1999; Choi and How, 2010b). When the system dynamics are linear (and any perturbations to the system or to the measurements are Gaussian), the posterior covariance given a sequence of measurements can be easily and accurately computed using the Kalman filter (EKF). However, when the dynamics are nonlinear, the calculation of the posterior must be approximate. In particular, when the dynamics are chaotic, complex, and of large-scale, Monte-Carlo methods are often used to estimate these nonlinear systems (Furrer and Bengtsson, 2007). The ensemble Kalman filter (EnKF) is one such Monte-Carlo filtering algorithm that has seen intensive use in numerical weather prediction (NWP) and other environmental sensing applications (Ott et al., 2004).

Evaluating a set of candidate measurement plans requires a set of Monte-Carlo simulations as a means for predicting the posterior covariance for each plan (Choi and How, 2010b). This process can be computationally demanding for large-scale nonlinear systems because producing an accurate prediction typically requires many samples for predicting the mean and covariance of the system state (Choi et al., 2008). Furthermore, additional measurements, which remain unknown during the planning time, will affect the

evolution of the covariance; we need the ability to revise the plan quickly upon receiving new measurements. Therefore, an efficient way of predicting future covariances given a set of measurements is essential to tractable planning of informative paths in large-scale nonlinear systems.

In this paper, we present the trajectory planning of a mobile sensor as a means for improving numerical weather predictions. Our primary contribution is to show that the planning process can be made substantially more efficient by replacing the Monte-Carlo simulation with an explicit model of covariance dynamics learned using a time-series regression. The regressed model is orders of magnitude faster to evaluate within the trajectory optimization. We also show that its accuracy of covariance prediction is better than other possible approximations such as linearization or partial use of the Monte-Carlo samples.

2. System Models and Forecasting

This work considers a large-scale dynamical system over a grid in which each cell represents a small region of the overall system and is associated with state variables representing physical quantities (e.g., temperature, pressure, concentration, wind speed) in the corresponding region. Without loss of generality, it is assumed that only one state variable is associated with a cell. The dynamics of the state variables are described by differential equations (Lorenz and Emanuel, 1998).

$$\dot{s}_{t,i} = f_i(\mathbf{s}_t), \, i \in \{1, 2, ..., n_s\} \tag{1}$$

where $S_{t,i} \in R$ denotes the state variable at the i-th grid point at time t, and n_S is the total number of grid points. We denote all state variables by the state vector $S_t \in R^{n_s}$. The dynamics f_i: $R^{n_s} \mapsto R$ are given by an arbitrary nonlinear function.

Knowing the current state of a system st, we can use it as the initial condition for the dynamics in Eq. (1), to compute a forecast of the system state in the future. However, we rarely know the true state of the system, but must take multiple (noisy) measurements of the system state, and use a filtering process to estimate $\mathbf{s}_t$ from these the measurements of the system and our knowledge of the system dynamics. The measurements $\mathbf{z}_t$ at time t are modeled as

$$\mathbf{z}_t = \mathbf{h}(\mathbf{s}_t, \mathbf{w}_t) \tag{2}$$

where $\mathbf{h}$ is the observation function, mapping state variables and sensing noise $\mathbf{w}_t$ to the measurements. In many sensing problems, especially in environmental sensing, the measurements are direct observations of the state variables subject to additive Gaussian sensing noise. In this case, the

measurement at the i-th grid point is given by

$$\mathbf{z}_t = s_{t,i} + w_{t,i} \tag{3}$$

where $w_{t,i} \sim N(0, W_i)$ and $\mathrm{E}[w_{t,i}\, w_{t,j}] = 0$, $i \neq j$.

In order to mitigate the effect of measurement noise, a probabilistic estimate $P(\mathbf{s}_t)$ of the state variables is typically computed using standard filtering algorithms. In many cases, the measurements are regularly taken at intervals, and recursive filtering algorithms that do not require access to the complete history of measurements are used. Standard recursive filtering algorithms consist of two steps.

Prediction: $P(\mathbf{s}_t | \mathbf{z}_{0:t}) \rightarrow P(\mathbf{s}_{t+1} | \mathbf{z}_{0:t})$ (4)

Update: $P(\mathbf{s}_{t+1} | \mathbf{z}_{0:t}) \rightarrow P(\mathbf{s}_{t+1} | \mathbf{z}_{0:t+1})$ (5)

where the prediction step gives the posterior distribution due to the dynamics, following the system dynamics $\mathbf{f} \triangleq [f_1, ..., f_{n_s}]^T$ and the update step incorporates the information from new measurements.

If $P(\mathbf{s}_t)$ is Gaussian, the first (mean) and the second (covariance) moments are sufficient statistics for describing the conditional distributions. For this case, the mean μ_t provides the best estimate of the state variables and the covariance Σ_t represents the uncertainty of the estimates. In order to distinguish the two distributions after prediction and update, we denote the two moments after prediction by μ^f and Σ^f, and after update by μ^a and Σ^a, following the convention of the weather community (Ott et al., 2004) where f denotes *forecast* and a denotes *analysis*.

When $P(\mathbf{s}_t)$ is non-Gaussian, the first and the second moments are used to approximately describe the distribution. Given the state estimate $P(\mathbf{s}_t | \mathbf{z}_{0:t})$, the T-timestep forecast $P(\mathbf{s}_{t+T} | \mathbf{z}_{0:t})$ can be made by performing only the prediction steps for T timesteps. The mean of the distribution μ_{t+T} then gives the forecast of the state variables and the covariance Σ_{t+T} gives the uncertainty of the forecast. As expected, the accuracy of the prediction may decrease with longer T.

2.1 Ensemble forecasting

The EnKF (Evensen and Van Leeuwen, 1996) (and it variants [Ott et al., 2004; Whitaker and Hamill, 2002]) is a Monte-Carlo (ensemble) version of the extended EKF. Monte-Carlo methods are often used to estimate nonlinear systems, especially when the system dynamics are chaotic, complex, and large-scale (Furrer and Bengtsson, 2007). The EnKF is one such Monte-Carlo filtering algorithm that has seen intensive use in numerical weather prediction (NWP) and other environmental sensing applications (Ott et al., 2004), since it typically represents the nonlinear features of the complex large-scale system, and mitigate the

computational burden of linearizing the nonlinear dynamics and keeping track of a large covariance matrix (Evensen and Van Leeuwen, 1996; Whitaker and Hamill, 2002), compared to the EKF.

In the EnKF, a set of possible state variable values is maintained in an ensemble matrix $\mathbf{S} \in \mathrm{R}^{n_s \times n_E}$ in which each column of the matrix is an ensemble member, that is, a possible state:

$$\mathbf{S} = [\mathbf{s}^1, \mathbf{s}^2, ..., \mathbf{s}^{n_E}] \tag{6}$$

where n_E is the size of the ensemble (number of Monte-Carlo samples). The prediction step corresponds to the nonlinear dynamics integration of an individual ensemble member

$$\mathbf{s}_{t+1}^i = \mathbf{s}_t^i + \int_t^{t+1} \dot{\mathbf{s}}\, d\tau, \quad \forall\, i \in \{1,...,n_E\} \tag{7}$$

Note that the dynamics integration of an ensemble member involves the integration of n_S variables; the computation time is $O(n_s \times n_s)$. The propagation of the ensemble members approximates the propagation of all statistical moments of the distribution via the nonlinear dynamics (and the more samples used in the simulation, the better the approximation). To compute the mean and covariance of the distribution, the EnKF uses the ensemble mean $\hat{\mathbf{s}}$ and perturbation ensemble matrix $\tilde{\mathbf{S}}$, defined as

$$\hat{\mathbf{s}} = \sum_{k \in \{1,...,n_E\}} (\mathbf{s}^k / n_E), \quad \tilde{\mathbf{S}} \equiv \gamma(\mathbf{S} - [\, \mathbf{s}, \cdots, \mathbf{s}\,]) \tag{8}$$

where $\gamma > 1$ is an inflation factor used to avoid underestimation of the covariance (Whitaker and Hamill, 2002), so that

$$\mu^f \approx \hat{\mathbf{s}}, \quad \Sigma^f \approx \tilde{\mathbf{S}}\tilde{\mathbf{S}}^T / (n_E - 1) \tag{9}$$

The measurement update step is similar to that of the EKF, where μ^f and Σ^f given by Equation 9 are used to calculate the Kalman gain matrix Kt. Each ensemble member is then updated according to

$$\mathbf{s}_t^i = \mathbf{s}_t^i + K_t(\mathbf{z}_t - H\mu_t^f), \quad \forall\, i \in \{1,...,n_E\} \tag{10}$$

A complete derivation of the EnKF is outside the scope of this paper; a full explanation of this nonlinear estimation method can be found in Whitaker and Hamill (2002).

3. Informative Path Planning

A significant problem existing in current operational systems for environmental sensing is that the measurements are unevenly distributed. Even with a perfect system dynamics model, the forecast of the system may degrade greatly for forecasts made far into the future, as the poor state estimate

propagates to other regions over multiple steps of forecasting (Morss et al., 2001). While additional measurements of the initial state estimate will clearly improve the forecast, the number of resources available to take measurements, such as unmanned aerial vehicle (UAV)-borne sensors, is finite. Furthermore, measurements at different locations and times possess different amounts of information due to the spatio-temporal correlations of the state variables. To make the most effective use of finite resources, we must consider the value of the possible measurements in terms of prediction performance (Berliner et al., 1999; Choi and How, 2010a, b; Morss et al., 2001), and choose to measure the state variables that maximize the prediction accuracy.

3.1 Adaptive targeting

Adaptive observation targeting has proven to be an important variant for informative path planning. Researchers have maintained great interest in adaptive observation targeting within the context of numerical weather prediction in which the figure of merit is forecast performance at a verification region (Choi and How, 2010b). For example, a goal may be established to take measurements over the Pacific to maximize the forecast accuracy for California, the verification region. The verification region is usually a small subset of the overall system; we denote the cells in this region by V.

There are also three important times in the weather targeting problem:

- Planning time $T_p \in Z$: the planning starts at T_p.
- Forecast time $T_f \in Z$: the time at which the last measurement will be taken, and a forecast will be generated. The entire planning, execution and forecasting process must be completed by T_f.
- Verification time $Tv \in Z$: the time at which the forecast will be tested. For instance, a 4-day forecast made at 9th September 2008 (T_f) will be verified at 13th September 2008 (T_v).

We can write the planning problem formally as

$$\mathbf{p}^* \in \ \arg\min_{\mathbf{p} \in \rho} J(\Sigma^f_{T_v}(V)) \tag{11}$$

subject to $\Sigma^f_{t+1} = F(\mu^a_t, \Sigma^a_t), \ \forall t[T_p\text{-}1, \ T_v\text{-}1] \cap Z$
$\quad\quad\quad\quad \Sigma^a_t = M(\mu^a_t, \Sigma^f_t, p_t), \ \forall t[T_p, \ T_v] \cap Z$
$\quad\quad\quad\quad \Sigma^a_t = \text{given}$

where $\Sigma(V)$ is the sub-block covariance of the cells in the verification region V, $\mathbf{p} \equiv \{p1, p2, ..., p_K\} \in Z^k$ is a sequence of cells that a sensor platform will visit over the time window $[T_p, T_f]$. $F(\cdot)$ is the covariance dynamics function that provides the posterior distribution after the prediction step. Note that

this function is not known in most cases; the EnKF simulates it by Monte-Carlo simulation and the EKF approximates it with linearization of the system dynamics. $M(\cdot, p)$ denotes the measurement update through the measurement taken at location p. The measurement taken after the forecast time T_f is typically not of great concern (Bishop et al., 2001). The set P is the feasible set of $\mathbf{p}$ that satisfies certain constraints such as motion of the sensing platforms. $J(\cdot)$ is a measure of uncertainty, and $\mathbf{p}^*$ is the optimal path which minimizes the uncertainty measure. This can be easily extended to a multi-agent problem where each agent chooses a path, and the collective information gain on V by these agents is optimized. Typically, trace or entropy of the covariance matrix are used as the measures of uncertainty (Berliner et al., 1999; Choi and How, 2010a). We use trace in this work, but the work can be generalized to entropy. Additionally, while we can design a sequence of future sensing points based on the knowledge available at the current decision time T_p, the evolution of covariance Σ requires the actual values of measurement in the future (i.e., the actual values of $z_{tk}(p_k)$), which are not available at T_p. This is an unavoidable source of the approximation to the covariance propagation and the planning algorithm may need to quickly cope with changes in the state of the system due to new measurements.

Note that the covariance dynamics function $F(\cdot)$, which represents the prediction of the covariance, is unknown in most cases; the EnKF simulates this function through Monte-Carlo simulation. Similarly, $M(\cdot, p)$ denotes the measurement update through the measurement taken at location p. For the prediction step, computing the covariance dynamics requires a long series of complex nonlinear integrations of each Monte-Carlo sample, as in Eq. (7). The forecast computation can be arbitrarily slow when more samples are used to improve the simulation of the covariance dynamics (Furrer and Bengtsson, 2007; Houtekamer and Mitchell, 2001). In the next section, we suggest that the direct learning of the input-output relationship of the Monte-Carlo simulation can provide fast and accurate approximations of covariance dynamics.

3.2 Local path planning

The informative path planning problem in a dynamic system is inherently an NP-hard problem (Choi and How, 2010b; Krause et al., 2008), as every possible plan sequence must be evaluated while the number of possible sequences exponentially grow with the length of the planning horizon. Given the large size of the domains under consideration, a multi-scale planning approach may be desired. While finding the global path is computationally infeasible, finding the best local path within a small region may be feasible.

Furthermore, there are some planning schemes available, which utilize locally optimal paths that make better choices at higher levels (Choi and How, 2010a). The local path planning task entails choosing the most informative path to move in between two waypoints, maximally reducing the uncertainty in the region between the waypoints. This decision making can also be addressed with the formulation in Eq. (11) by setting $T_f = T_v$ and taking the verification region V as the local sub-region between the start point and the endpoint.

4. Covariance Dynamics Learning

We wish to approximate the *covariance* dynamics, which we model as a nonlinear function F so that

$$F(P(\mathbf{s}_t)) = \Sigma_{t+1} \tag{13}$$

where $P(\mathbf{s}_t)$ is the probability distribution of st described by its statistical moments

$$P(\mathbf{s}_t) \approx P\{\mu, \Sigma_t\} \tag{14}$$

where μ_t is the first moment, the mean vector, and Σ_t is the second moment, the covariance matrix. The covariance dynamics F is unknown in analytic form, but is induced by the system dynamics, and is approximated by Monte-Carlo simulation in the EnKF.

Since we do not know the exact form of $F(\cdot)$, we propose to learn an approximator (equivalently, model or function) $\hat{F}$ from past input-output samples of the Monte-Carlo simulations of the covariance dynamics F, called *training* samples in machine learning literature (Evgeniou et al., 2000). Additionally, we will derive important *features* from the input, transforming the input data into real-valued vectors of the same length $\mathbf{x}_t$,

$$F(P(\mathbf{s}_t)) \approx \hat{F}(\mathbf{x}_t) \tag{15}$$

Specifically, regression is used in order to learn a model with continuous-valued output. Regression learns a target function g given a finite number of training samples; inputs $X=(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n)$ and outputs $\mathbf{y}=(y_1, y_2, ..., y_n)$, where $g(\mathbf{x}_i)=y_i \in \mathbb{R}, \ \forall i\{1, ..., n\}$. The learned model $\hat{g}$ is expected to have the property

$$\hat{g}(\mathbf{x}_i)=y_i+e_i \text{ where } \mathrm{E}[e_i]=0, \ \forall i, \text{ and } E[e_i e_j]=0 \text{ for } i \neq j \tag{16}$$

The learning approach aims to achieve accuracy by relying on nonlinear features of the input to represent the complex nonlinear function F, which is contrasted to the linearization of the EKF that assumes local linearity of system dynamics. The other important advantage in using the regression

approach is that the accuracy of the covariance dynamics approximation increases with the number of Monte-Carlo samples used in the original simulation, without the penalty of the increasing computation time. The reason is that the original simulation will become increasingly slow with more Monte-Carlo samples, but will provide a better approximation of the covariance dynamics. This results in training samples that will act as more accurate examples of the true covariance dynamics F. Still, the computation time for learning $\hat{F}$ and the prediction time of using the learned model $\hat{F}$ will remain nearly the same, being independent of the accuracy of the training samples (Evgeniou et al., 2000).

In principle, we can learn a predictor of any function of the future covariance; for example, a direct mapping from the current covariance to the trace of the verification sub-block of the covariance after k timesteps with measurements taken along path $\mathbf{p}$,

$$\hat{F}_K^V(P(\mathbf{s}_t), \mathbf{p}) \approx \mathrm{trace}\{\Sigma_{t+k}^f(V)\}. \tag{17}$$

However, we argue that learning the one-step approximator $\hat{F}$ is the best approach. One has to consider the fact that as the target function becomes more complicated, more training samples and sophisticated features may be required. While generating more training samples only requires more Monte-Carlo simulations, finding sophisticated features requires finding a proper transformation of the original features through many trial and error investigations (Guyon and Elisseeff, 2003). In addition, learning common functions which can be applied for different paths may be preferred. Having learned the one-step approximator $\hat{F}$, we can use the approximator to evaluate any path combined with a given measurement update function $\mathrm{M}(\cdot, \cdot)$. In order to predict the covariance multiple timesteps into the future, we can use recursive prediction, in which the output of at time t+1 is used as input at time $t+2$ and so on. The learning of $\hat{F}$ will be relatively easier than that of $\hat{F}_K^V$ or other more complicated functions.

4.1 Feature selection: autoregressive time-series mode

The input to the true covariance dynamics F is the current covariance and other statistical moments. In learning $\hat{F}$, we select features in order to avoid overfitting and manage computational cost. However, learning with many features creates additional problems since many of the features are noisy or irrelevant; in other words, selecting the most relevant features has led to more efficient learning (Guyon and Elisseeff, 2003). We use the idea of state-space reconstruction (Box et al., 1994; Sauer et al., 1991), a standard methodology

for learning nonlinear dynamical systems. The methodology uses time-series observations of a system variable in order to learn the dynamics of the system. The time-series of the observed (specified) variable contains information of the unobserved (unspecified) variables; thus, learning the dynamics only through observed variables is possible.

Following this idea, we used the time-series of the covariance as the only features for learning the covariance dynamics; the covariance dynamics were modeled with autoregressive (AR) time-series features (Box et al., 1994) so that

$$F(P(\mathbf{s}_t)) = \Sigma_{t+1} \approx \widehat{F}(\Sigma_{(t-d+1):t}) \tag{18}$$

where $\Sigma_{t_1:t_1+d} = \{\Sigma_{t_1}, \Sigma_{t_1+1}, ..., \Sigma_{t_1+d}\}$ and d is the degree, or the length, of time-series used as features. As the degree d increased, the resulting model was able to represent more complicated functions. We used cross-validation (Evgeniou et al., 2000) to choose d so that the model possessed sufficient complexity necessary for representing the covariance dynamics, but did not overfit to the training samples.

There are two types of covariances at time t; the forecast (or predicted) covariance Σ_t^f and analysis (or updated) covariance Σ_t^a. We simply extend the definition of $\Sigma_{t_1:t_2}$ to

$$\Sigma_{t_1:t_2} = \left\{ \Sigma_{t_1}^f, \Sigma_{t_1}^a, \Sigma_{t_1+1}^f, \Sigma_{t_1+1}^a ... \Sigma_{t_2}^f, \Sigma_{t_2}^a \right\} \tag{19}$$

In this way, the effect of the covariance dynamics and the measurement update will be learned altogether. The learning approach is compared to the other methods in Fig. 1.

4.1.1 Function decomposition

Because $\widehat{F}$ takes a series of d covariances as input in which each size is $n_s \times n_s$, and generates a future covariance of size $n_s \times n_s$, it is clearly a multidimensional function of the form

$$\widehat{F} : \mathbb{R}^{n_s \times n_s \times d} \mapsto \mathbb{R}^{n_s \times n_s}. \tag{20}$$
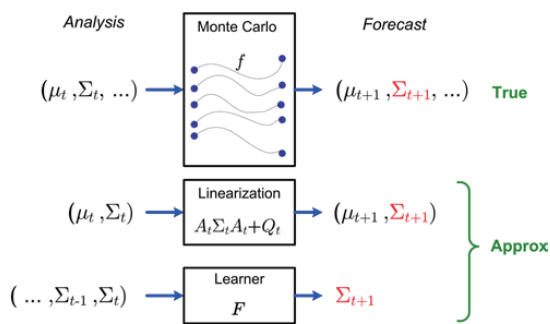

Fig. 1. The block diagram of the covariance dynamics approximation scheme in filtering algorithms and the time-series learning approach.

To simplify the learning problem, we decompose $\widehat{F}$ into $O(n_s \times n_s)$ sub-functions

$$\widehat{F}^{(i,j)}(\Sigma_{(t-d+1):t}(i,j)) \approx \Sigma_{t+1}(i,j) \text{ where } i \leq j \text{ and } i,j\{1, ...N_s\} \tag{21}$$

$$\widehat{F}^{(j,i)} = F^{(i,j)} \text{ if } i \neq j \tag{22}$$

The principle of the state-space construction states that this learning problem is solvable as the dynamics of a variable can be learned from the time-series observations of the variable. In addition to the simplicity of learning, decomposition takes into account the locality of the covariance dynamics that the system dynamics f_i can be different for each cell i and that the statistics at each cell are affected by the distribution of the measurement network.

4.1.2. Fast recursive prediction

In many physical systems, the spatially neighboring variables are strongly coupled, and thus their covariances are also expected to be coupled (Furrer and Bengtsson, 2007). Instead of just using the AR features $\Sigma_{(t-d+1):t}(i, j)$ in learning $\widehat{F}^{(i,j)}$, we might also use the time-series of the spatially neighboring cells, $\Sigma_{(t-d+1):t}(\mathbf{N}_L(i), \mathbf{N}_L(j))$, where $\mathbf{N}_L(i)=\{k|\mathbf{k-i}|^2 \leq L\}$ for some constant L and $\mathbf{i}$ represents the location vector of cell i. However, possible accuracy improvement is negated by the computational burden of the recursive prediction that is needed for multi-step prediction.

In recursive prediction, all features needed for the next step prediction must also be predicted from the current step. Suppose we want to predict a covariance entry $\Sigma_{t+2}(i, j)$ from Σ_t. In using neighboring features as input to the prediction, we must also predict the sub-block covariance $\hat{\Sigma}_{t+1}(\mathbf{N}_L(i), \mathbf{N}_L(j))$. The entries of $\hat{\Sigma}_{t+1}(\mathbf{N}_L(i), \mathbf{N}_L(j))$ have to be predicted using their own spatial neighbors from Σ_t. It is easy to see that the number of the entries to be predicted increases with the number of steps in the prediction; the prediction of $\Sigma_{t+3}(i, j)$ requires $\hat{\Sigma}_{t+2}(\mathbf{N}_L(i), \mathbf{N}_L(j))$ and so on. With the use of AR features alone, we only have to know the past values of one covariance entry to predict the next value of the entry. (Editor's Note: Very well written paragraph)

4.2 Model selection: parametric model learning

In this work, we decided to learn a parametric model of the covariance dynamics, representing $\hat{F}^{(i,j)}$ by a linear function of the input $\mathbf{x} = \Sigma_{t-d+1:t}(i, j)$

$$\Sigma_{t+1}(i, j) \approx \hat{F}^{(i,j)}(\mathbf{x}) = \beta_0^{(i,j)} + \beta_1^{(i,j)}\Sigma_{t-d+1}(i,j) + \beta_2^{(i,j)}\Sigma_{t-d+2}(i,j) \\ +...+\beta_d^{(i,j)}\Sigma_t(i,j) \tag{23}$$

where $\beta^{(i,j)} \triangleq \{\beta_0^{(i,j)}, \beta_1^{(i,j)}, ..., \beta_d^{(i,j)}\}$ are the coefficients to be determined by the training process. The linear function in

Eq. (23) models nonlinear or more complicated functions of the original input **x** by incorporating nonlinear functions of the input features **x**. For instance, we can transform $\mathbf{x}=(x_1, x_2, \dots x_m)$ to $\mathbf{x}'=(x_1, x_1^2, x_2, x_2^2, \dots x_m, x_m^2)$ by adding squared terms of original features; this is called basis expansion. Then, a linear function of **x**′ will be a nonlinear or quadratic function of original feature **x**. Basis expansion and the use of higher degree time-series features are the two strategies that we employed to learn the nonlinear covariance dynamics.

The choice of learning a parametric model is to enable faster prediction during the operation. Once the coefficients $\beta^{(i,j)}$ are found through a training process, the computation time of a parametric model is linear in the number of features. This is contrasted to popular non-parametric models such as support vector machines or Gaussian processes, in which prediction time is proportional to the number of training samples (Evgeniou et al., 2000). We want to learn a global model that is valid for the entire sample space enabling the use of the model in operation without frequent updates. To learn a global model, we expected to use as many training samples as possible, and the initial experiments showed that the learned covariance dynamics was only valid locally if trained with a small number of training samples (Park, 2008). The use of nonparametric models may be prohibitive in this case.

4.2.1 Regularized least squares regression

Learning of $\hat{F}^{(i,j)}$ entails finding the coefficients $\beta^{(i, j)}$ through minimizing some loss function of prediction errors, $e_k= \hat{F}^{(i,j)}(\mathbf{x}_k)-y_k$, $k\in\{1, \dots, n\}$. We used regularization to learn $\hat{F}^{(i,j)}$, which did not overfit to the training samples (Evgeniou et al., 2000). Overfitting models may offer zero prediction error by perfectly fitting the training samples; however, such a procedure poorly predicts a new example **x***, where $\mathbf{x}^*\neq\mathbf{x}_k$, $k\in\{1, \dots, n\}$.

Regularization provides extra information from domain knowledge by placing a prior over the regression coefficients β; for example, the L^2-norm of the coefficients $|\beta|^2$ is preferred to be small, as larger $|\beta|^2$ indicating that the learned function is less smooth and may be the result of overfitting the training samples. The regularized least squares (RLS) algorithm minimizes the sum of the squared-error of the training samples and a regularization penalty,

$$\hat{\beta} = \arg\min_{\beta} \| (X\beta - \mathbf{y}) \|^2 +\lambda\beta^T\beta . \tag{24}$$

where λ is regularization parameter which controls the contribution of the L2-norm of regression coefficients β to the total loss function; a larger value of λ encourages smaller $|\beta|^2$. To minimize Eq. (24), we set the derivative of Eq. (24)

with respect to β to 0, and get

$$\hat{\beta} = (X^{\mathrm{T}} X + \lambda I)^{-1} X^{\mathrm{T}}\mathbf{y}. \tag{25}$$

We find $\hat{\beta}^{(i,j)}$ for each approximator $\hat{F}^{(i,j)}$ using the RLS algorithm, while $\lambda^{(i,j)}$, the degree of the time-series d, and the proper basis expansion are found by cross-validation. For instance, k-fold cross validation divides the training samples into k sets and the training process is repeated for k times. At each trial, only k-1 sets are used for the training and the other one is used for calculating the prediction error. The prediction error is averaged for the k trials. The model with the lowest average prediction error usually provides the best model that does not overfit to the specific training samples (Evgeniou et al., 2000). There are $O(n_s\times n_s)$ models to be learned and the total training time is $O(n_s^2\times n^2)$ where n is the number of training samples. Training is performed permanently and can be done offline, while UAVs use the learned models with fast prediction time during operation.

4.3 Path selection with learned covariance dynamics

Once we have learned the models $\hat{F}$, we can predict the uncertainty of the system after taking a series of observations, using the learned models and the measurement update function. Specifically, we want the prediction for a region of interest, such as a verification region. Let R represent the cells that correspond to the region, then the number of system variables contained in those cells, i.e., $|R|$, is typically significantly smaller than the number of state variables, n_S. The computational complexity of calculating the sub-block forecast covariance $\Sigma_t(R)$ is given in Table 1. It is shown that only the AR prediction has a computation time independent of the system size n_S. For the full propagation method, the size of the ensemble should grow with the number of states to achieve sufficient level of estimation performance.

Full-propagation needs to integrate each ensemble member, as in Eq. (7). Each ensemble represents nS state

Table 1. The computational complexity of the algorithms for the one-step prediction (and calculating the forecast covariance). C_{int} is the nontrivial cost of nonlinear dynamics integration of one state variable; n_E is the ensemble size; n_S is the number of state variables in the system. n_E has to be $\Omega(n_s^2)$(Furrer and Bengtsson, 2007). $|R|$ is the size of the region of interest. d is the size of the features for autoregressive (AR) models. $d\ll n_E$, $|R|\ll n_S$ for many path planning cases so that AR prediction becomes significantly faster

	Complexity
Full propagation	$\Omega(C_{int}n_En_S+n_E|R|^2)$
Linearization	$O(C_{int}n_S^3)$
AR prediction	$O(d|R|^2)$

variables, and the integration of one ensemble member takes $O(C_{int}n_s)$ where C_{int} is the integration cost of a variable using certain methods, such as RK4 method. Furthermore, a larger system should be estimated with a larger ensemble; n_E has to be $\Omega(n_S^2)$(Furrer and Bengtsson, 2007). Therefore, the total computation time is $\Omega(C_{int}n_En_S)$. The calculation of the covariance matrix incurs additional $\Omega(n_E|R|^2)$ computational effort because this calculation involves computing inner products of n_E-dimensional vector for $|R|^2$ times. The linearization calculates the Jacobian matrix around the current mean estimate μ_{t-1} and applies it to $\Sigma_{t-1}(R)$, which yields the computation time $O(C_{int}n_S^2)$.

However, the learned model calculates each entry of $\Sigma_t(R)$ using the time-series $\Sigma_{t-d:t-1}(R)$. An entry of the predicted covariance matrix is a linear sum of d features, so the computation time is $O(d|R|^2)$ as there are $|R|^2$ entries. The computation time is independent of nS and this results in a great computation reduction given $|R| \ll n_S$. The degree of time-series d represents the complexity of the AR model, which may grow as the complexity of the system dynamics grows. However, the system size n_S does not have a direct impact on d. Also, d is independent of n_E; a large ensemble set can be used for better estimation of a system, but this will only provide better training samples of the true covariance dynamics.

For the measurement update function, we can use a localized measurement update for further computation reduction, which ignores spurious correlations between two variables that are physically far from each other (Houtekamer

and Mitchell, 2001); for a measurement at location p, only the physical neighbors $\mathbf{N}_L(p)$ are updated where L represents the maximum distance that an observation can affect. The use of the covariance dynamics model and the localized measurement update function facilitates path planning without maintaining the full covariance matrix. Specifically, we only need the sub-block covariance of the cells in the verification (or local) region V, on path $\mathbf{p}$, and the local neighbors of $\mathbf{p}$ denoted by $\mathbf{N}_L(p)$. Given the uncertainty at a future time, problems regarding path planning become trivial in regards to choosing the path providing maximum uncertainty reduction, as in Algorithm 1 (Fig. 2).

5. Numerical Experiments

In this work, we used the Lorenz-2003 weather model (Lorenz, 2005) for all experiments and for method validation. The Lorenz models are known for their nonlinear and chaotic behavior, and have been used extensively in the validation of adaptive observation strategies for weather prediction (Choi and How, 2010a; Leutbecher, 2003; Lorenz and Emanuel, 1998).

5.1 Lorenz-2003 model

The Lorenz-2003 model (Choi and How, 2010b) is an extended model of the well-known Lorenz-95 model (Lorenz and Emanuel, 1998) that addresses multi-scale features of the weather dynamics in addition to the basic aspects of the weather motion such as energy dissipation, advection, and external forcing. We used the two-dimensional Lorenz-2003 model, representing the mid-latitude region (20-70 deg) of the northern hemisphere. There are $L_{on}=36\alpha$ longitudinal and $L_{at}=8\beta+1$ latitudinal grids in Lorenz models. We used $\alpha=\beta=2$; resulting in $72 \times 17 = 1,224$ state variables. The length-scale of the Lorenz models was proportional to the inverse of α and β in each direction: the grid size for $\alpha=\beta=2$ amounts to 347 km $\times$ 347 km. The time-scale of the Lorenz models was such that 1 time unit was equivalent to 5 days in real time; the duration of 0.01 time units (or 1.2 hours) was equivalent to 1 (discrete) timestep in the further discussions.

We used a two-dimensional index (i, j) in which i denotes the West-to-East grid index and j denotes the South-to-North grid index; $s_{(i,j)}$ is the state variable of (i, j)-th grid. The dynamic equations governing the state variables is

$$\dot{s}(i,j) = -\xi_{(i-4,j)}\xi_{(i-2,j)} + (1/3)\sum_{k=-1}^{+1}\xi_{(i-2+k,j)}s(i+k,j)$$
$$- (2/3)\eta_{(i,j-4)}\eta_{(i,j-2)} + (2/9)\sum_{k=-1}^{+1}\eta_{(i,j-2+k)}s(i,j+k) \quad (26)$$
$$- s(i,j) + s_0, \quad i \in \{1,...,72\}, \quad j \in \{1,...,17\}$$

Algorithm 1. Path selection algorithm
Input: Initial (analysis) covariance Σ_0^a and learned models $\hat{F}$
for all paths $\mathbf{p}^k$ **do**   Get the sub-block covariance $\Sigma_0^a(R)$, $R \triangleq U\{V,|\mathbf{p}^k, \mathbf{N}_L(\mathbf{p}^k)\}$

for all locations $l_t \in \mathbf{p}^k$, $t \in \{1, ..., T_f\}$ **do**
  Propagate elements of $\Sigma_{t-1}^a(R)$ according to the learned
  models $\hat{F}(i, j)$, $(i, j) \in R$
  Perform (localized) measurement update at location l_t
  and get $\Sigma_t^a(R)$

end for
Propagate elements of $\Sigma_{T_f}(V)$ for more steps to get $\Sigma_{T_v}(V)$, $T_f \leq T_v$
end for
Return the path $\mathbf{p}^*$ with minimum trace of $\Sigma_{T_v}(V)$

Fig. 2. Algorithm 1: Path selection algorithm.

where $\xi(i, j)=\Sigma(1/3)\Sigma_{k=-1}^{+1}s(i+k, \ j)$, $\eta(i, \ j)=(1/3)\Sigma_{k=-1}^{+1}s(i, \ j+k)$. The equations contain quadratic, linear, and constant terms representing advection, dissipation, and external forcing. The dynamics in Eq. (26) are subject to cyclic boundary conditions in the longitudinal direction: $s_{(i+72, \ j)}=s_{(i-72, \ j)}=s(i, \ j)$ and a constant advection condition $s_{(i \ 0)}=\cdots=s_{(i, \ -1)}=3$, $s_{(i, \ 18)}=\cdots=s_{(i, \ 18)}=0$ is applied in the latitudinal direction.

5.2 Covariance dynamics learning results

Table 2 shows the prediction performance of the learned covariance dynamics using different features, while the RLS algorithm was used for all features. The one-step prediction of covariance using the learned models was compared to that of using the true covariance dynamics from Monte-Carlo simulations. The metric used for the prediction performance was the normalized mean squared error (NMSE), which is the MSE normalized by the total variance of the predicted variable. The MSE can be small, even with poor learned models, when the predicted variable exhibited very small variance. The NMSE does not have this problem. The models were trained with a set of training samples and tested on a separate test set in order to measure how well the learned models generalized to the new samples. The values in Table 2 show the prediction errors on the test set.

The AR features containing different degrees, the quadratic and cubic basis expansions, and the spatial neighbors features were compared. Based on the comparison, we clearly observed that the covariance dynamics was better modeled with more complicated models using higher-degree AR features and the basis expansion of the original AR features. The spatial neighbor features also helped prediction performance, but were less useful than the basis expansion of the AR features. It also has the disadvantage of the slow recursive prediction. Note that the number of features increased with the basis expansion; if the original features were AR features with degree d, the full quadratic expansion increases the number of features to $O(d^2)$ and the full cubic expansion increases it to $O(d^3)$. For instance, the

models with full quadratic expansion can be 20 times slower than the models with original features when d = 20. Thus, we did not use the full basis expansion and added only a few interaction terms to the model.

In accordance with the results, we chose d = 20 as the degree of the AR features in the final models. The use of a higher value of d, $d > 20$, resulted in overfitting. The prediction error (test error) started to grow, while the prediction time slowed down as the value of d increased. The value of d is linear to the number of features used in the model. However, it is possible to see the prediction error reduction with higher values of d and with more training samples. Again, the specific choice of d and the proper basis expansion must be selected by a model selection procedure, such as cross-validation or using a separate test set, and we chose d = 20 which had the best test error. We used the quadratic basis expansion for its similar performance to the cubic basis expansion. However, in contrast to the cubic basis expansion, the quadratic basis expansion generates faster models.

We compared the accuracy of the learned covariance with other approximation methods and the true covariance dynamics from Monte-Carlo simulations. The partial propagation uses only a random fraction of the original Monte-Carlo samples, which gives a constant factor speed-up as the prediction time is linear to the number of the Monte-Carlo samples; if we use 10% of the samples, we get a speed-up of about 10 times that of the original simulation. Note that, however, the computation time of the partial propagation scales with the size of the dynamical system, unlike the learned dynamics case, as in Table 1.

The prediction of the trace of the sub-block covariance $\Sigma(V)$ for a same path is tested in Table 3, where V is an 11 × 11 verification region. The verification time was 20 steps after the forecast time and a sequence of 11 measurements was taken before the forecast. The predictions using the learned covariance dynamics were significantly better than those from the linearized model. The partial propagation also performed worse than the learned dynamics in terms

Table 2. Time-series regression result (NMSE) (3,200 training samples and 800 test samples); the error bars represent ±2σ

Model	Degree of time-series (d)		
	5	10	20
AR features ($D = 0$)			
RLS (linear)	$2.8\times10^{-3} \pm 4.6\times10^{-3}$	$3.2\times10^{-6} \pm 1.3\times10^{-5}$	$5.4\times10^{-7} \pm 2.6\times10^{-6}$
RLS (quadratic)	$2.8\times10^{-3} \pm 5.1\times10^{-3}$	$9.2\times10^{-7} \pm 4.0\times10^{-6}$	$2.3\times10^{-7} \pm 1.1\times10^{-6}$
RLS (cubic)	$3.0\times10^{-3} \pm 5.9\times10^{-3}$	$4.6\times10^{-7} \pm 2.0\times10^{-6}$	$2.3\times10^{-7} \pm 9.1\times10^{-7}$
AR + spatial neighbors features ($D = 1$)			
RLS (original)	$2.7\times10^{-3} \pm 4.6\times10^{-3}$	$3.6\times10^{-6} \pm 1.1\times10^{-5}$	$4.2\times10^{-7} \pm 2.4\times10^{-6}$

AR: autoregressive, RLS: regularized least squares.

Table 3. The prediction result (mean absolute error) of the trace of posterior distribution after path executions (20 step forecast after 11 step planning)

	The size of ensemble	
	1,224	6,120
Learned	0.0669 ± 0.0657	0.3032 ± 0.0882
Linearization	0.1502 ± 0.1439	0.5801 ± 0.1483
Partial (10%) propagation	0.0718 ± 0.0450	0.4834 ± 0.0531

Table 4. Average computation time for path planning using different methods: A total of 51 paths were evaluated for the informative forecasting of Lorenz-2003 model using (a) full propagation of ensemble Kalman filter with $n_E = 1,224$, (b) linearization, and (c) AR prediction. (Pentium-4 3.0 Ghz Dual Core was used)

Full propagation	1,984.48 s
Linearization	21.79 s
AR(20) prediction	6.81 s

AR: autoregressive.

Fig. 4. Local planning result: accumulated difference of trace between autoregressive (20) model prediction, greedy and linearization method of 5-timestep local planning in Lorenz-2003 model.

Fig. 5. Verification region targeting result: accumulated difference of trace (11-timestep planning and 20-timestep forecast) in Lorenz-2003 model.
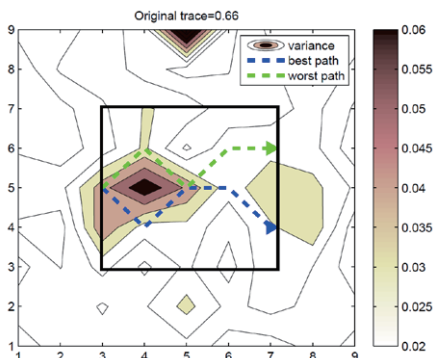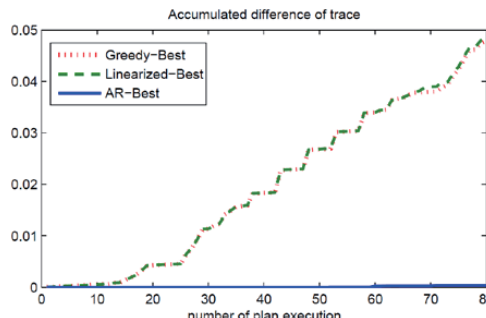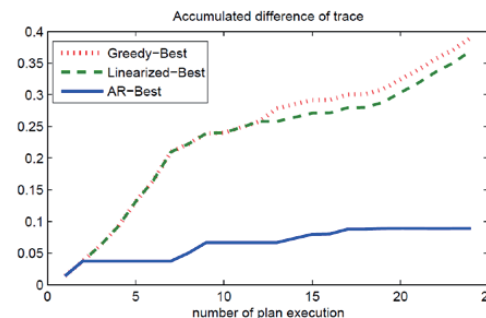

Fig. 3. Original state of the ensemble Kalman filter at planning time. The best path and worst path are shown. The rectangle represents the local region of interest that we plan to minimize the uncertainty (trace of covariance).

of mean prediction error, especially with a bigger original ensemble, but had smaller prediction error variance.

The path selection requires the accurate relative ranking of the candidate paths. For that purpose, both the prediction error (bias) and the variance should be small. Roughly, the bias may cause consistent error in the ranking and the variance causes the occasional error in the ranking. We show the path selection result in the next section.

5.3 Path planning results

Given the ability to predict the change in covariance of the EnKF efficiently, we can now use the learned function to identify sensor trajectories that maximize information gain for some region of interest. We assume that a UAV moves from one cell to another of the eight neighboring cells at certain timesteps, taking a measurement at every cell it visits. The system also has a fixed observation network, where about 10 percent of the system is covered by the routine observation network that provides measurements in a regular interval (5 timesteps for this work). We tested two path planning scenarios: adaptive targeting and local path planning. The measure of information gain is the change in the trace of the R sub-block of the covariance $\Sigma(R)$. R is either a local region or a verification region depending on the planning problem.

A sample scenario of the local path planning is given in Fig. 3. We considered the case where R is a 5×5 local region and the planning horizon is 5. Given the initial state of the EnKF, an agent plans to observe a sequence of 5 locations, one for every timestep, before arriving at the end point. We fixed the start point and used three choices for the end point, thus there were a total of 51 paths in this case. The best and worst paths are illustrated in Fig. 3. In terms of information gain, the best path decreased the trace of the covariance of

335

the region by 15% while the worst path actually increased the trace 1.5%. The average reduction of the uncertainty was 8.3% with standard deviation of 4.6% for 51 paths.

Under the same scenario, we tested the path selection ability of different strategies. The baseline greedy strategy is to choose the path with maximum uncertainty reduction in the current covariance using the measurements; it does not propagate the uncertainty in time and but performs measurement updates on the current covariance. The results of local path selection is shown in Fig. 4. The AR strategy makes almost no mistakes in this case, and the greedy strategy performs similarly to the linearization method. The actual computation time in the experiment is shown in Table 4. The AR prediction was much faster than full propagation and was also faster than linearization. This computational advantage was also theoretically proven with a lower time complexity of the algorithm as in Table 1. The computational advantage will be even greater in larger dynamical systems with bigger Ns than the Lorenz-2003 model.

The other scenario was the targeting problem in the 11×11 region near the verification region in Fig. 1, and the reduction of the trace of 20-step forecast (the verification time was 20 steps after the forecast time) was evaluated. There are about 8,000 possible paths, but we randomly selected 100 paths for our experiments; we may not have chosen the true best path, as it may be one of the other 7,900 paths, but the relative performance of the strategies should not be affected by this random selection. The result of the path selection using different strategies is shown in Fig. 5. The prediction result of the learned covariance dynamics enabled the selection of the path close to the true best path of the EnKF.

6. Conclusions

This paper presented a learning method that improves computational efficiency in the optimal design of sensor trajectories in a large-scale dynamic environment. A time-series regression method based on autoregressive features and a regularized least-squares algorithm was proposed to learn a predictive covariance dynamics model. It was empirically verified that the proposed learning mechanism successfully approximated the covariance dynamics and significantly reduced the computational cost in calculating the information gain for each candidate measurement path. In addition, numerical results with a simplified weather forecasting example suggested that path planning based on the presented learning method outperformed other approximate planning schemes such as linearization and partial Monte-Carlo propagation.

References

Berliner, L. M., Lu, Z. Q., and Snyder, C. (1999). Statistical design for adaptive weather observations. *Journal of the Atmospheric Sciences*, 56, 2536-2552.

Bishop, C. H., Etherton, B. J., and Majumdar, S. J. (2001). Adaptive sampling with the ensemble transform Kalman filter Part I: Theoretical aspects. *Monthly Weather Review*, 129, 420-436.

Box, G. E. P., Jenkins, G. M., and Reinsel, G. C. (1994). *Time Series Analysis: Forecasting and Control*. 3rd ed. Englewood Cliffs, NJ: Prentice Hall.

Choi, H. L., How, J., and Hansen, J. (2008). Algorithm and sensitivity analysis of information-theoretic ensemble-based observation targeting. *Proceedings of AMS Annual Meeting*, New Orleans, LA.

Choi, H. L. and How, J. P. (2010a). Coordinated targeting of mobile sensor networks for ensemble forecast improvement. *IEEE Sensors Journal*, in press.

Choi, H. L. and How, J. P. (2010b). Efficient targeting of sensor networks for large-scale systems. *IEEE Transactions on Control Systems Technology*, *in press*.

Evensen, G. and Van Leeuwen, P. J. (1996). Assimilation of geosat altimeter data for the agulhas current using the ensemble kalman filter with a quasigeostrophic model. *Monthly Weather Review*, 124, 85-96.

Evgeniou, T., Pontil, M., and Poggio, T. (2000). Regularization Networks and Support Vector Machines. *Advances in Computational Mathematics*, 13, 1-50.

Furrer, R. and Bengtsson, T. (2007). Estimation of high-dimensional prior and posterior covariance matrices in Kalman filter variants. *Journal of Multivariate Analysis*, 98, 227-255.

Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3, 1157-1182.

Houtekamer, P. L. and Mitchell, H. L. (2001). A sequential ensemble Kalman filter for atmospheric data assimilation. *Monthly Weather Review*, 129, 123-137.

Krause, A., Singh, A., and Guestrin, C. (2008). Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, 9, 235-284.

Leutbecher, M. (2003). A reduced rank estimate of forecast error variance changes due to intermittent modifications of the observing network. *Journal of the Atmospheric Sciences*, 60, 729-742.

Lorenz, E. N. (2005). Designing chaotic models. *Journal of the Atmospheric Sciences*, 62, 1574-1587.

Lorenz, E. N. and Emanuel, K. A. (1998). Optimal sites for supplementary weather observations: Simulation with a small model. *Journal of the Atmospheric Sciences*, 55, 399-414.

Morss, R. E., Emanuel, K. A., and Snyder, C. (2001). Idealized adaptive observation strategies for improving numerical weather prediction. *Journal of the Atmospheric Sciences*, 58, 210-232.

Ott, E., Hunt, B. R., Szunyogh, I., Zimin, A. V., Kostelich, E. J., Corazza, M., Kalnay, E., Patil, D. J., and Yorke, J. A. (2004). A local ensemble Kalman filter for atmospheric data assimilation. *Tellus, Series A: Dynamic Meteorology and Oceanography*, 56, 415-428.

Park, S. (2008). *Learning for Informative Path Planning.* PhD Thesis, Massachusetts Institute of Technology.

Sauer, T., Yorke, J. A., and Casdagli, M. (1991). Embedology. *Journal of Statistical Physics*, 65, 579-616.

Whitaker, J. S. and Hamill, T. M. (2002). Ensemble data assimilation without perturbed observations. *Monthly Weather Review*, 130, 1913-1924.