AIAA Guidance, Navigation, and Control Conference
10 - 13 August 2009, Chicago, Illinois

AIAA 2009-6116

Learning covariance dynamics for path planning of UAV sensors in a large-scale dynamic environment

Sooho Park*, Han-Lim Choi†, Nicholas Roy‡ and Jonathan P. How§

This work addresses the problem of trajectory planning for UAV sensors taking measurements of a large nonlinear system to improve estimation and prediction of such a system. The lack of perfect knowledge of the global system state typically requires probabilistic state estimation. The goal is therefore to find trajectories such that the measurements along each trajectory minimize the expected error of the predicted state of the system some time into the future. The considerable nonlinearity of the dynamics governing these systems necessitates the use of computationally costly Monte-Carlo estimation techniques to update the state distribution over time. This computational burden renders planning 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 model of the covariance dynamics learned using nonlinear time-series regression. The use of autoregressive (AR) time-series features with the regularized least squares (RLS) algorithm enables 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 both terms of accuracy and speed, with examples of simplified weather forecasting.

I. Introduction

Many natural phenomena change rapidly. As a result, regular measurements must be taken to determine the current state of different natural systems such as the weather, vegetation, animal herds, coral reef health, etc. Without sufficiently frequent measurements, the difference between the estimated state of these dynamic systems and the true state can grow arbitrarily large. Sensor networks have been used with success in many real-world domains in order to automate the measurement process, but in these domains we are usually limited in how the sensors can be deployed. Complete sensor coverage of any real-world domain is impractical for reasons of 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 the system.

The fast moving dynamics of natural domains present two challenges. The state dynamics are strongly coupled (part of the system can often have far-reaching influence on the rest of the system) and also chaotic (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 known sufficiently accurately, the

*Computer Science and Artificial Intelligence Laboratory, MIT, dreamneo@csail.mit.edu
†Aerospace Controls Laboratory, MIT, hanlimc@mit.edu, Member AIAA.
‡Computer Science and Artificial Intelligence Laboratory, MIT, nickroy@mit.edu
§Aerospace Controls Laboratory, MIT, jhow@mit.edu, Associate Fellow AIAA

accuracy of the estimated state of the whole system is significantly reduced, as is our ability to make predictions about the future.

Additional measurements can be used to improve the accuracy of state predictions. But again, given finite resources, only a subset of the state can be measured with additional sensors. If these sensors are mobile, then we would like to choose 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, then the sequence of measurements that minimizes some norm on the covariance is usually preferred.

The path selection problem (or "informative path planning" problem) is straightforward to solve if we can efficiently compute the posterior covariance after taking a sequence of the measurements [1, 2]. 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. 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 [3]. 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 [4]

Evaluating a set of candidate measurement plans requires a set of Monte-Carlo simulations to predict the posterior covariance for each plan [2]. This process can be computationally demanding, in particular for large-scale nonlinear systems, because accurate prediction typically requires many samples to predict the mean and covariance of the system state [5]. Furthermore, additional measurements, that are 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 describe the problem of planning the trajectory of a mobile sensor to improve numerical weather prediction. 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 the covariance dynamics learned using 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.

II. System Models and Forecasting

This work considers a large-scale dynamical system over a grid, where 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 [6]

$$\dot{s}_{t,i} = f_i(\mathbf{s}_t), \quad i \in \{1, \cdots, N_s\} \tag{1}$$

where $s_{t,i} \in \Re$ 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 $\mathbf{s}_t \in \Re^{N_s}$. The dynamics $f_i : \Re^{N_s} \mapsto \Re$ are given by an arbitrary nonlinear function.

Knowing the current state of a system $\mathbf{s}_t$, we can use it as the initial condition for the dynamics in Equation 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

$$z_{t,} = s_{t,i} + w_{t,i}, \tag{3}$$

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

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.

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

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

where the prediction step gives the posterior distribution due to the dynamics, following the system dynamics $\mathbf{f} \triangleq [f_1, \cdots, f_{N_s}]^T$ 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. In this case, the mean μ_t gives the best estimate of the state variables and the covariance Σ_t represents the uncertainty of the estimates. 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 [4] where f denotes *forecast* and a denotes *analysis*.

Even when $P(\mathbf{s}_t)$ is non-Gaussian, the first and the second moments are used for approximately describing 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.

II.A. Ensemble Forecasting

The Ensemble Kalman Filter (EnKF) [7] (and it variants [4, 8]) is a Monte-Carlo (ensemble) version of the extended Kalman Filter (EKF). When system dynamics are chaotic, complex, and of large-scale, Monte-Carlo methods are often used to estimate these nonlinear systems [3]. The EnKF is one such Monte-Carlo filtering algorithm that has seen intensive use in numerical weather prediction (NWP) and other environmental sensing applications [4], 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 [7, 8], compared to the EKF.

In the EnKF, a set of possible values of the state variables is maintained in an ensemble matrix $\mathbf{S} \in \Re^{N_s \times N_e}$ where each column of the matrix is an ensemble member, that is, a possible state:

$$\mathbf{S} = \begin{bmatrix} \mathbf{s}^1 & \mathbf{s}^2 & \dots & \mathbf{s}^{N_e} \end{bmatrix} \tag{6}$$

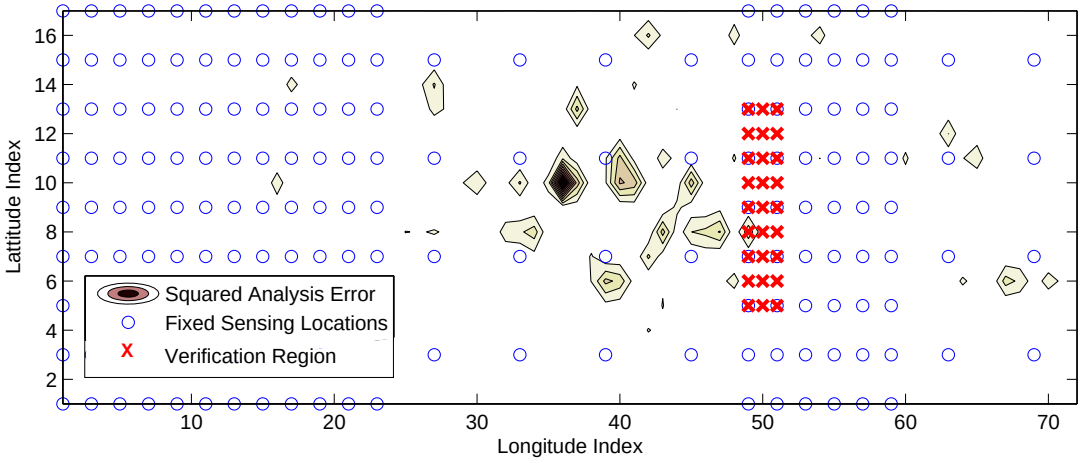

Figure 1. The squared estimation error of an EnKF estimate with 6120 ensemble members for the Lorenz-2003 model.

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

$$\mathbf{s}_{t+1}^i = \mathbf{s}_t^i + \int_t^{t+1} \dot{\mathbf{s}} d\tau \text{ for all } i \in \{1, ..., N_e\} \tag{7}$$

Note that the dynamics integration of an ensemble member involves the integration of N_s variables; the computation time is $O(N_s \times N_e)$. 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 $\widehat{\mathbf{s}}$ and perturbation ensemble matrix $\widetilde{\mathbf{S}}$, defined as

$$\widehat{\mathbf{s}} = \frac{1}{N_e} \sum_{k \in [1, N_e]} \mathbf{s}^k, \quad \widetilde{\mathbf{S}} \equiv \gamma(\mathbf{S} - [\, \widehat{\mathbf{s}}, \cdots, \widehat{\mathbf{s}} \,]) \tag{8}$$

where $\gamma > 1$ is an inflation factor used to avoid underestimation of the covariance [8], so that

$$\mu^f \approx \widehat{\mathbf{s}}, \quad \Sigma^f \approx \frac{1}{N_e - 1} \widetilde{\mathbf{S}}\, \widetilde{\mathbf{S}}^T \tag{9}$$

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 K_t. Each ensemble member is then updated according to

$$\mathbf{s}_t^i = \mathbf{s}_t^i + K_t(\mathbf{z}_t - H\mu_t^f) \text{ for all } i \in \{1, ..., N_e\} \tag{10}$$

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 [8].

III. Informative Path Planning

A significant problem with current operational systems for environmental sensing is that the measurements are unevenly distributed. Figure 1 shows an example of this case taken from a weather simulation domain, where measurements over large ocean areas are sparse compared to measurements over land. In Figure 1, regular measurements are taken at about 10% of the entire area, densely distributed on the left side and the right side, at constant intervals while the middle area is modeled as the Pacific, where measurements are sparse. Even with a perfect model of the

system dynamics, 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 [9].

While additional measurements of the initial state estimate will clearly improve the forecast, the number of resources available to take measurements, such as UAV-borne sensors, is finite. Furthermore, measurements at different place and time have different amount 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 [1, 2, 9, 10], and choose to measure the state variables that maximize the prediction accuracy.

III.A. Adaptive Targeting

An important variant of the informative path planning problem of great interest in the context of numerical weather prediction is adaptive observation targeting, where the figure of merit is forecast performance at a verification region [2]; for example, the goal may be to take measurements over the Pacific to maximize the accuracy of the forecast 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: the planning starts at T_p.
- Forecast time T_f: 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 T_v: the time at which the forecast will be tested. For instance, a 4-day forecast made at 9th Sep 2008 (T_f) will be verified at 13th Sep 2008 (T_v).

We can write the planning problem formally as

$$\begin{aligned} & \mathbf{p}^\star = \arg\min_{\mathbf{p}\in\mathcal{P}} \mathcal{J}(\Sigma^f_{T_v}(V)) \\ \text{subject to} \quad & \Sigma^f_{t+1} = F(\mu^a_t, \Sigma^a_t), \quad \forall t \in [T_p - 1, T_v - 1] \\ & \Sigma^a_t = M(\mu^a_t, \Sigma^f_t, p_t), \quad \forall t \in [T_p, T_f] \\ & \Sigma^a_{T_p} = \text{given}, \end{aligned} \tag{11}$$

where $\Sigma(V)$ is the sub-block covariance of the cells in the verification region V, $\mathbf{p} \equiv \{p_1, p_2, \cdots, p_K\} \in \mathbb{Z}^K$ 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 gives 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 out of concern [11]. The set $\mathcal{P}$ is the feasible set of $\mathbf{p}$ that satisfies certain constraints such as motion of the sensing platforms. $\mathcal{J}(\cdot)$ is a measure of uncertainty, and $\mathbf{p}^\star$ 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 measure of uncertainty [1, 10]. 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 $z_{t_k}(p_k)$), 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 cope with changes in the state of the system due to new measurements quickly.

Note that the covariance dynamics function $F(\cdot)$, which represents the prediction of the covariance, is unknown in most cases; the EnKF simulates it by 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 [3, 12]. In the next section, we suggest that the direct learning of the input-output relationship of the Monte-Carlo simulation can provide the fast and accurate approximation of the covariance dynamics.

III.B. Local Path Planning

The informative path planning problem in a dynamical system is inherently a NP-hard problem [2, 13], as every possible plan sequence must be evaluated while the number of possible sequences grow exponentially with the length of planning horizon. Given the large size of domains to be considered, a multi-scale planning approach may be desired. While finding the global path is computationally infeasible, finding the local best path within a small region may be feasible. Furthermore, there are some planning schemes available, which can utilize locally optimal paths to make better choices at higher levels [10]. The local path planning task is to choose the most informative path to move between two waypoints, maximally reducing the uncertainty in the region between the waypoints. We denote this region of interest as L. Then, the goal of local planning is to find the optimal path $\mathbf{p}^*$ that minimizes the uncertainty of the region L.

$$\begin{aligned} & \mathbf{p}^\star = \arg\min_{\mathbf{p}\in\mathcal{P}} \mathcal{J}(\Sigma^f_{T_f}(L)) \\ \text{subject to} \quad & \Sigma^f_{t+1} = F(\mu^a_t, \Sigma^a_t), \quad \forall t \in [T_p - 1, T_f - 1] \\ & \Sigma^a_t = M(\mu^a_t, \Sigma^f_t, p_t), \quad \forall t \in [T_p, T_f] \\ & \Sigma^a_{T_p} = \text{given}, \end{aligned} \tag{12}$$

where $\Sigma(L)$ is the L sub-block of the covariance.

IV. 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 $\mathbf{s}_t$ described by its statistical moments

$$P(\mathbf{s}_t) \approxeq \{\mu_t, \Sigma_t\} \tag{14}$$

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) $\widehat{F}$ from past input-output samples of the Monte Carlo simulations of the covariance dynamics F, called *training* samples in machine learning literature [14]. Additionally, we

will derive important *features* from the input, transforming the input data into real-valued vectors of the same length $\mathbf{x}_t$,

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

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 $g(\mathbf{x}_i) = y_i \in \Re$ for all $i \in \{1, ...n\}$. The learned model $\widehat{g}$ is expected to have the property

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

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 to 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 be more accurate examples of the true covariance dynamics F. Still, the computation time for learning $\widehat{F}$ and the prediction time of using the learned model $\widehat{F}$ will stay similar, being independent of the accuracy of the training samples [14].

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 $\mathbf{p}$,

$$\widehat{F}_k^V(P(\mathbf{s}_t), \mathbf{p}) \approx \text{tr}\{\Sigma_{t+k}^f(V)\}. \tag{17}$$

However, we argue that learning the one-step approximator $\widehat{F}$ 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 [15]. In addition, learning common functions which can be applied for different paths may be preferred. Having learned the one-step approximator $\widehat{F}$, we can use it for the evaluation of any path combined with a given measurement update function $M(\cdot, \cdot)$. 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 $\widehat{F}$ will be relatively easier than that of $\widehat{F}_k^V$ or other more complicated functions.

IV.A. Feature Selection: Autoregressive Time-series Model

The input to the true covariance dynamics F is the current covariance and other statistical moments. In learning $\widehat{F}$, 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 are irrelevant; selecting the most relevant features have leads to more efficient learning [15]. We use the idea of state-space reconstruction [16, 17], a standard methodology for learning nonlinear dynamical systems, in reasoning about the right features. The methodology uses the 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 the information of the unobserved (unspecified) variables, so that learning the dynamics through only observed variables is possible.

Following this idea, we use the time-series of the covariance as the only features in learning the covariance dynamics; the covariance dynamics are modeled with autoregressive (AR) time-series

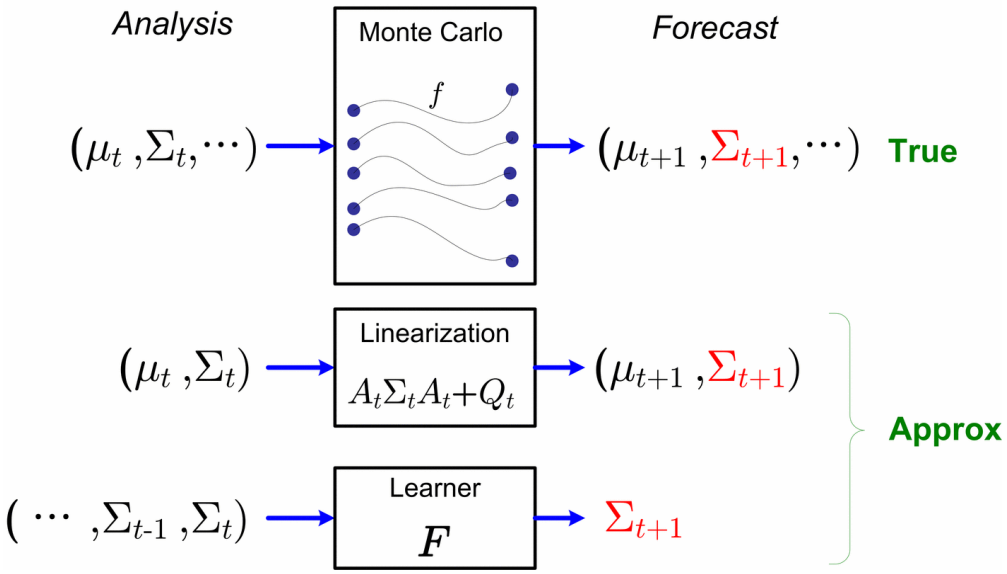

Figure 2. The block diagram of covariance dynamics approximation scheme in filtering algorithms and the time-series learning approach.

features [17] 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 increases, the resulting model can represent more complicated functions. We use cross-validation [14] to choose d so that the model has the enough complexity to represent the covariance dynamics, but does 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

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

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 Figure 2.

IV.A.1. Function decomposition

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

$$\widehat{F} : \Re^{N_s \times N_s \times d} \rightarrow \Re^{N_s \times N_s}. \tag{20}$$

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

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

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

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. Besides the simplicity of learning, the 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.

IV.A.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 [3]. 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 $\mathbf{N}_L(i) = \{k||\mathbf{k}-\mathbf{i}|^2 \leq L\}$ for some constant L and $\mathbf{i}$ represents the location vector of cell i. However, the possible accuracy improvement is weighed down 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 $\widehat{\Sigma}_{t+1}(\mathbf{N}_L(i),\mathbf{N}_L(j))$. The entries of $\widehat{\Sigma}_{t+1}(\mathbf{N}_L(i),\mathbf{N}_L(j))$ have to 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 $\widehat{\Sigma}_{t+2}(\mathbf{N}_L(i),\mathbf{N}_L(j))$ 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.

IV.A.3. Fast sub-block covariance prediction

Another advantage of using AR features is that faster prediction of sub-blocks of the covariance can be carried out independently of the entire covariance. The computation time of a covariance sub-block using the learned dynamics is $O(R \times d)$ where R is the size of the sub-covariance, and d is the size of the features (the degree of time-series features), which should be orders of magnitude smaller than N_s in large systems. In contrast, using a filter based on linearization requires evaluating the Jacobian matrix first, which takes $O(C_{int} \times N_s)$, where C_{int} is the integration cost of the dynamics. Being able to predict the covariance of sub-blocks independently is very useful for a local planning or a verification region prediction.

IV.B. Model Selection: Parametric Model Learning

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

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

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) can model nonlinear or more complicated functions of the original input $\mathbf{x}$ by incorporating nonlinear functions of the input features $\mathbf{x}$. For instance, we can transform $\mathbf{x} = (x_1, x_2, ...x_m)$ to $\mathbf{x}' = (x_1, x_1^2, x_2, x_2^2, ...x_m, x_m^2)$ by adding squared terms of original features; this is called *basis expansion*. Then, a linear function of $\mathbf{x}'$ will be a nonlinear or quadratic function of original feature $\mathbf{x}$. Basis expansion and the use of higher degree time-series features are the two strategies that we use 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 the popular non-parametric models such as Support Vector Machines (SVM) or Gaussian Processes (GP), whose prediction time is proportional to the number of training samples [14]. We want to learn a global model that is valid for the entire sample space, so that the model can be used in operation without frequent updates. To learn a global model, we expect 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 [18]. The use of nonparametric models may be prohibitive in this case.

IV.B.1. Regularized Least Squares Regression

The learning of $\widehat{F}^{(i,j)}$ is to find the coefficients $\beta^{(i,j)}$ through minimizing some loss function of prediction errors, $e_k = \widehat{F}^{(i,j)}(\mathbf{x}_k) - y_k, k \in \{1, ..., n\}$. We use *regularization* to learn $\widehat{F}^{(i,j)}$, which does not overfit to the training samples [14]. Overfitting models may have zero prediction error by perfectly fitting the training samples, but generalize poorly in predicting a new example $\mathbf{x}^*$, where $\mathbf{x}^* \neq \mathbf{x}_k, k \in [1, n]$.

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 $|\beta|^2$ means 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,

$$\widehat{\beta} = \underset{\beta}{\operatorname{argmin}} \quad ||(X\beta - \mathbf{y})||^2 + \lambda \beta^T \beta \tag{24}$$

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 (24), we set the derivative of (24) with respect to β to 0, and get

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

We find $\widehat{\beta}^{(i,j)}$ for each approximator $\widehat{F}^{(i,j)}$ using the RLS algorithm, while $\lambda^{(i,j)}$, the degree of 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 gives the best model which does not overfit to the specific training samples [14]. 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. The training is performed once for all and this can be done offline, while UAVs use the learned models with fast prediction time during the operation.

IV.C. Path selection with learned covariance dynamics

Once we have learned the models $\widehat{F}$, 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 corresponding to the region, then the number of system variables contained in those cells,

	Complexity
Full Propagation	$\Omega(C_{int} N_e N_s + N_e \lvert R \rvert^2)$
Linearization	$O(C_{int} N_s^2)$
AR Prediction	$O(d \lvert R \rvert^2)$

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 $\Omega(N_s^2)$ [3]. $|R|$ is the size of the region of interest. d is the size of the features for AR models. $d \ll N_e$, $|R| \ll N_s$ in many path planning case so that AR prediction is significantly faster.

i.e., $|R|$, is typically significant 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 the computation time that is independent of the system size N_s. The nonlinear dynamics integration in both methods, and the size of ensemble that should grow large for estimating a large system for the full propagation method, are the reasons for the computation time that scales with the system size.

The full-propagation needs to integrate each ensemble member as in Eq. (7). Each ensemble represents N_s state 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 methods such as RK4 method. Furthermore, a larger system should be estimated with a larger ensemble; N_e has to be $\Omega(N_s^2)$ [3]. Therefore, the total computation time is $\Omega(C_{int}N_eN_s)$. The calculation of covariance matrix incurs additional $\Omega(N_e|R|^2)$ computational effort because it is done by computing inner products of N_e-dimensional vector for $|R|^2$ times. The linearization has to calculate the Jacobian matrix around the current mean estimate μ_{t-1} and apply 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 N_s 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 and this may grow with the complexity of the system dynamics, but 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 which are physically far from each other [12]; for a measurement at location p, only the physical neighbors $\mathbf{N}_L(p)$ are updated where L represent the maximum distance that an observation can affect. The use of the covariance dynamics model and the localized measurement update function enables the 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 $N_L(\mathbf{p})$. Given the uncertainty at future time, the path planning problem becomes trivial as to choose the path which gives the maximum uncertainty reduction, as in the algorithm 1.

Algorithm 1 Path Selection Algorithm

Input: Initial (analysis) covariance Σ_0^a and learned models $\widehat{F}$
for all paths $\mathbf{p}^k$ **do**
  Get the sub-block covariance $\Sigma_0^a(R)$, $R \triangleq \bigcup\{V, \mathbf{p}^k, \mathbf{N}_L(\mathbf{p}^k)\}$
  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 $\widehat{F}(i,j)$, $(i,j) \in R$
    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 $\Sigma_{T_v}(V)$

V. Numerical Experiments

In this work, we use the Lorenz-2003 weather model [19] 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 strategy for weather prediction [6, 10, 20].

V.A. Lorenz-2003 Model

The Lorenz-2003 model [2] is an extended model of the well-known Lorenz-95 model [6] that addresses multi-scale feature of the weather dynamics in addition to the basic aspects of the weather motion such as energy dissipation, advection, and external forcing. We use the two-dimensional Lorenz-2003 model, representing the mid-latitude region ($20^\circ - 70^\circ$) of the northern hemisphere. There are $L_{on} = 36\alpha$ longitudinal and $L_{at} = 8\beta + 1$ latitudinal grids in Lorenz models. We use $\alpha = \beta = 2$; then, there are $72 \times 17 = 1224$ state variables. The length-scale of the Lorenz models are proportional to the inverse of α and β in each direction: the grid size for $\alpha = \beta = 2$ amounts to 347 km × 347 km. The time-scale of the Lorenz models are such that 1 time units are equivalent to 5 days in real time; the duration of 0.01 time units (or 1.2 hrs) is equivalent to 1 (discrete) timestep in the further discussions.

We use a two-dimensional index (i, j) where 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

$$\begin{aligned}\dot{s}(i,j) = &-\xi_{(i-2\alpha,j)}\xi_{(i-\alpha,j)} + \tfrac{1}{2\alpha'+1}\textstyle\sum_{k=-\alpha'}^{+\alpha'} \xi_{(i-\alpha+k,j)}s(i+k,j) \\ &- \tfrac{2}{3}\eta_{(i,j-2\beta)}\eta_{(i,j-\beta)} + \tfrac{2/3}{2\beta'+1}\textstyle\sum_{k=-\beta'}^{+\beta'}\eta_{(i,j-\beta+k)}s(i,j+k) \\ &- s(i,j) + s_0, \qquad i = 1,\ldots,L_{on}, \quad j = 1,\ldots,L_{at}\end{aligned} \tag{26}$$

where

$$\xi_{(i,j)} = \tfrac{1}{2\alpha'+1}\textstyle\sum_{k=-\alpha'}^{+\alpha'} s(i+k,j), \quad \eta_{(i,j)} = \tfrac{1}{2\alpha'+1}\textstyle\sum_{k=-\beta'}^{+\beta'} s(i,j+k)$$

with $\alpha' = \lfloor \alpha/2 \rfloor$ and $\beta' = \lfloor \beta/2 \rfloor$. The equations contain quadratic, linear, and constant terms representing advection, dissipation, and external forcing. The dynamics of the (i, j)-th grid point depends on its longitudinal 2α-interval neighbors (and latitudinal 2β) through the advection terms, on itself by the dissipation term, and on the external forcing. The dynamics in (26) are subject to cyclic boundary conditions in longitudinal direction: $s_{(i+L_{on},j)} = s_{(i-L_{on},j)} = s_{(i,j)}$, and a constant advection condition $s_{(i,0)} = \cdots = x_{(i,-\lfloor\beta/2\rfloor)} = 3$, $s_{(i,L_{at}+1)} = \cdots = s_{(i,L_{at}+\lfloor\beta/2\rfloor)} = 0$ is applied in the latitudinal direction.

V.B. Covariance Dynamics Learning Results

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

The AR features with different degree, the quadratic and cubic basis expansions, and the spatial neighbors features were compared. It is clear from the result that the covariance dynamics was

better modeled with more complicated models using a higher-degree AR features and the basis expansion of the original AR features. The spatial neighbors features also helped the 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 increases with the basis expansion; if the original features is the AR features of 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.

According to the result, we choose $d = 20$ as the degree of the AR features in the final models. The use of higher $d > 20$ resulted in overfitting that the prediction error (test error) started growing, while the prediction time gets slower with higher d as the prediction time is linear to the number of features used in the model. However, it is possible to see the prediction error improve with higher d 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 choose $d = 20$ which had the best test error. We use the quadratic basis expansion as it had the similar performance to the cubic basis expansion but 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 MC samples; if we use 10% of the samples, we get the speed-up of about 10 times from 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 a 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 of the mean prediction error, especially with 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 the consistent error in the ranking and the variance cause the occasional error in the ranking. We show the path selection result in the next section.

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

Table 2. Time-series regression result (NMSE) (3200 training samples and 800 test samples). The error bars represent $\pm 2\sigma$.

	The size of ensemble	
	1224	6120
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 3. The prediction result (mean absolute error) of the trace of posterior distribution after path executions (20 step forecast after 11 step planning)

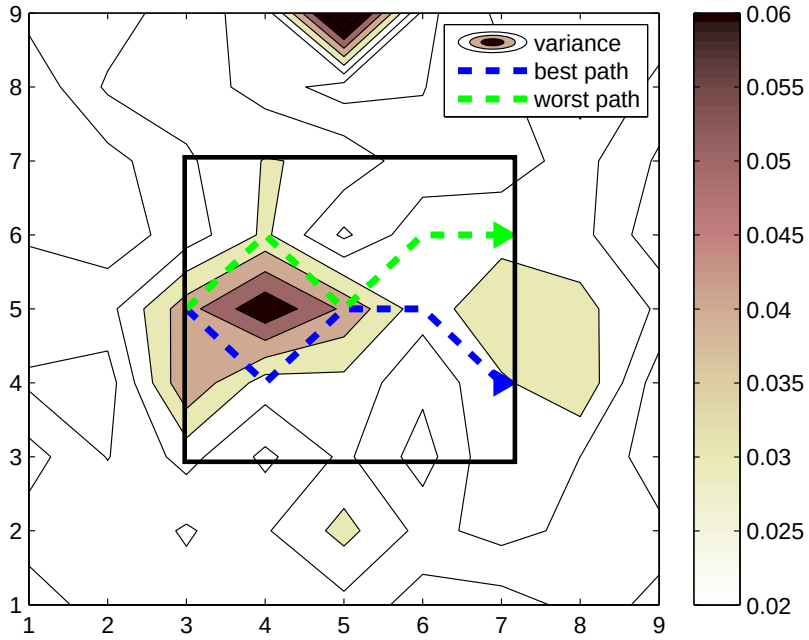

Figure 3. Original state of the EnKF at planning time. The best path and worst path are shown. The rectangle represent the local region of interest that we plan to minimize the uncertainty (trace of covariance).

V.C. 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 an UAV moves from one cell to one of the eight neighboring cells every timesteps, taking a measurement at every cell it visits. The system also has the fixed observation network shown in Figure 1, where about 10 percent of the system are covered by the routine observation network that provide measurements in a regular interval (5 timesteps for this work). We tested the 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 Figure 3. We considered the case where R is a 5x5 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 choice of paths in this case. The best and worst paths are illustrated given in Figure 3. In terms of information gain, the best path decreased the trace of the covariance of 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.

	Lorenz-2003 (N_e=1224)
Full Propagation	1984.48s
Linearization	21.79s
AR(20) Prediction	6.81s

Table 4. Average computation time for path planning for different methods. A total of 51 paths were evaluated using full propagation of EnKF, linearization and AR prediction. Full measurement update was used for full propagation and localized measurement update was used for linearization and AR prediction. (Pentium-4 3.0 Ghz Dual Core was used).

Under the same scenario, we test the path selection ability of different strategies. The baseline greedy strategy is to choose the path with the 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 result of local path selection is shown in Figure 4. The AR strategy makes almost no mistakes in this case, and the greedy strategy performs similar to the linearization method. The actual computation time in the experiment is shown in Table 4. The AR prediction is much faster than full propagation and also faster than linearization. This computational advantage was also theoretically proved with the lower time complexity of the algorithm as in Table 1. The computational advantage will be even greater in larger dynamical systems, with bigger N_s than the Lorenz-2003 model.

The other scenario is the targeting problem in the 11×11 region near the verification region in Figure 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 8000 possible paths, but we randomly selected 100 of them for the experiments; we may not choose the true best path, as it may be one of the other 7900 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 was shown in Figure 5. The prediction result of the learned covariance dynamics enabled the selection of the path close to the true best path of the EnKF.

VI. Conclusions

This paper presented a learning method to improve computational efficiency in optimal design of sensor trajectories in a large-scale dynamic environment. A time-series regression method based on autoregressive features and regularized least-squares algorithm is proposed to learn a predictive model of the covariance dynamics. It was empirically verified that the proposed learning mechanism successfully approximate the covariance dynamics and significantly reduce 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 outperforms other approximate planning schemes such as linearization and partial Monte-Carlo propagation.

Acknowledgment

This work is funded by NSF CNS-0540331 as part of the DDDAS program with Dr. Suhada Jayasuriya as the program manager. Sooho Park was partially funded by the Samsung Scholarship Foundation.

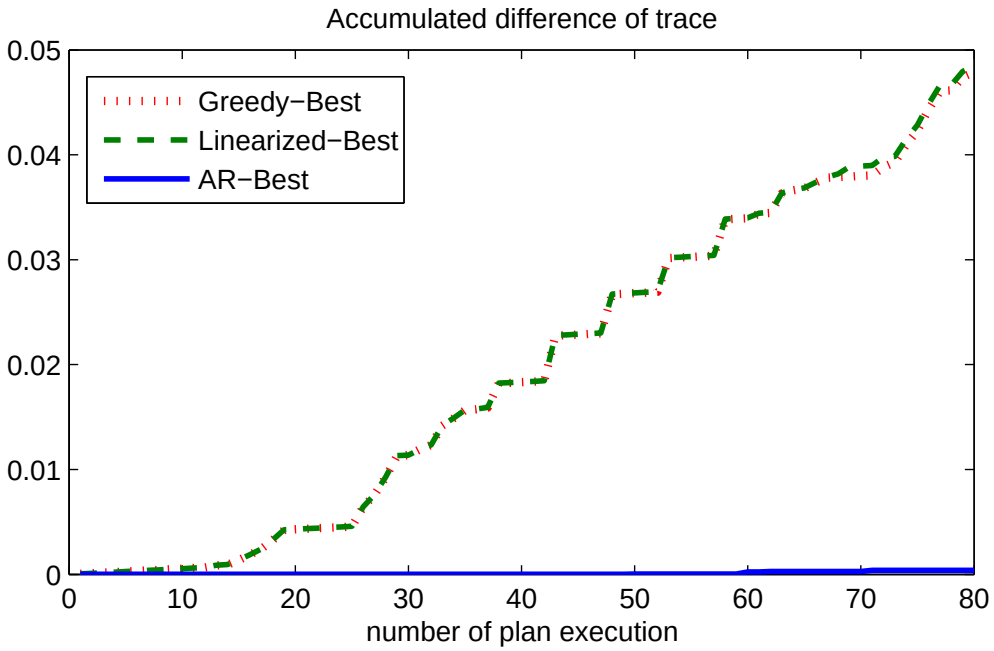

Figure 4. Local Planning Result: accumulated difference of trace between AR(20) model prediction, greedy and linearization method of 5-timestep local planning in Lorenz-2003 model.

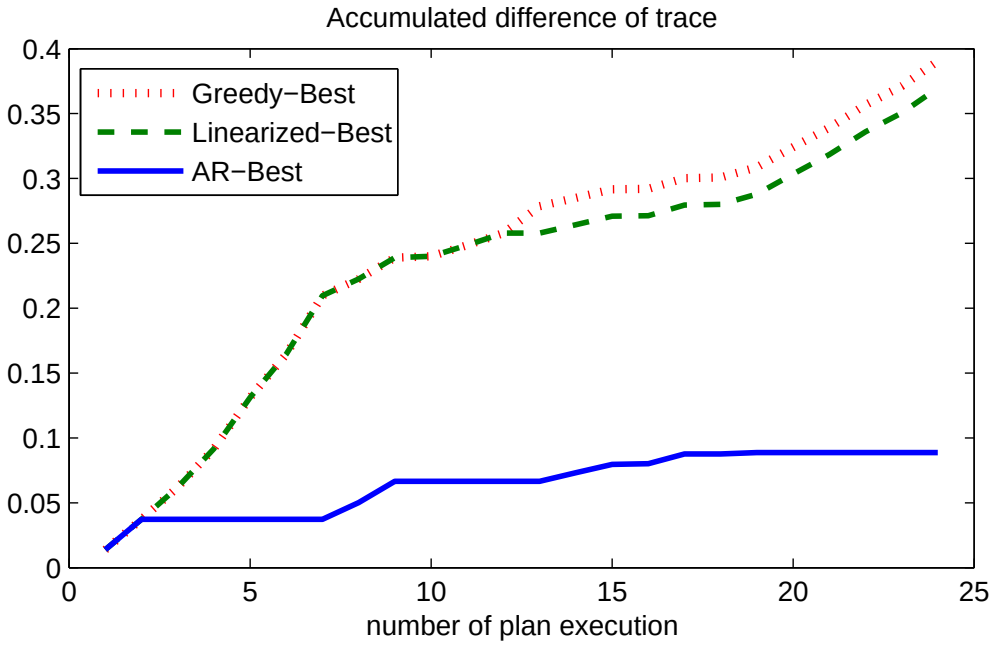

Figure 5. Verification Region Targeting Result: accumulated difference of trace (11-timestep planning and 20-timestep forecast) in Lorenz-2003 model.

References

[1] Berliner, L. M., Lu, Z. Q., and Snyder, C., "Statistical Design for Adaptive Weather Observations," *Journal of the Atmospheric Sciences*, Vol. 56, No. 15, 1999, pp. 2536–2552.

[2] Choi, H. L., How, J. P., and Hansen, J. A., "Ensemble-Based Adaptive Targeting of Mobile Sensor Networks," *American Control Conference, ACC'07.*, 2007.

[3] Furrer, R. and Bengtsson, T., "Estimation of high-dimensional prior and posterior covariance matrices in Kalman filter variants," *Journal of Multivariate Analysis*, Vol. 98, No. 2, 2007, pp. 227–255.

[4] Ott, E., Hunt, B. R., Szunyogh, I., Zimin, A. V., Kostelich, E. J., Corazza, M., Kalnay, E., Patil, D. J., and Yorke, J. A., "A local ensemble Kalman filter for atmospheric data assimilation," *Tellus A*, Vol. 56, No. 5, 2004, pp. 415–428.

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

[6] Lorenz, E. N. and Emanuel, K. A., "Optimal Sites for Supplementary Weather Observations: Simulation with a Small Model," *Journal of the Atmospheric Sciences*, Vol. 55, No. 3, 1998, pp. 399–414.

[7] Evensen, G. and van Leeuwen, P., "Assimilation of altimeter data for the Agulhas current using the ensemble Kalman filter with a quasigeostrophic model," *Monthly Weather Review*, Vol. 123, No. 1, 1996, pp. 85–96.

[8] Whitaker, J. and Hamill, H., "Ensemble Data Assimilation without Perturbed Observations," *Monthly Weather Review*, Vol. 130, No. 7, 2002, pp. 1913–1924.

[9] Morss, R., Emanuel, K., and Snyder, C., "Idealized Adaptive Observation Strategies for Improving Numerical Weather Prediction," *Journal of the Atmospheric Sciences*, Vol. 58, No. 2, 2001.

[10] Choi, H. L. and How, J. P., "A Multi-UAV Targeting Algorithm for Ensemble Forecast Improvement," *AIAA Guidance, Navigation, and Control Conference, GNC'07.*, 2007.

[11] Bishop, C. H., Etherton, B. J., and Majumdar, S. J., "Adaptive Sampling with the Ensemble Transform Kalman Filter. Part I: Theoretical Aspects," *Monthly Weather Review*, Vol. 129, No. 3, 2001, pp. 420436.

[12] Houtekamer, P. and Mitchell, H., "A Sequential Ensemble Kalman Filter for Atmospheric Data Assimilation," *Monthly Weather Review*, Vol. 129, No. 1, 2001, pp. 123–137.

[13] Guestrin, C., Krause, A., and Singh, A., "Near-Optimal Sensor Placements in Gaussian Processes," *Proceedings of the 22nd International Conference on Machine Learning*, Bonn, Germany, 2005.

[14] Evgeniou, T., Pontil, M., and Poggio, T., "Regularization Networks and Support Vector Machines," *Advances in Computational Mathematics*, Vol. 13, No. 1, 2000.

[15] Guyon, I., Elisseefi, A., and Kaelbling, L. P., "An Introduction to Variable and Feature Selection," *Journal of Machine Learning Research*, Vol. 3, No. 7-8, 2003, pp. 1157–1182.

[16] Sauer, T., Yorke, J. A., and Casdagli, M., "Embedology," *Journal of Statistical Physics*, Vol. 65, No. 3/4, 1991.

[17] Box, G. E. P. and Jenkins, G., *Time Series Analysis, Forecasting and Control*, Holden-Day, Incorporated, 1990.

[18] Park, S., *Learning for Informative Path Planning*, Master's thesis, MIT, September 2008.

[19] Lorenz, E. N., "Designing Chaotic Models," *Journal of the Atmospheric Sciences*, Vol. 62, No. 5, 2005, pp. 1574–1587.

[20] Leutbecher, M., "A Reduced Rank Estimate of Forecast Error Variance Changes due to Intermittent Modifications of the Observing Network," *Journal of the Atmospheric Sciences*, Vol. 60, No. 5, 2003, pp. 729–742.