Evolving Gaussian Processes and Kernel Observers for Learning and Control in Spatiotemporally Varying Domains

with applications in agriculture, weather monitoring, and fluid dynamics

Joshua E. Whitman, Harshal Maske, Hassan A. Kingravi, Girish Chowdhary,
POC email: girishc@illinois.edu

Introduction

Monitoring and modeling of large-scale stochastic phenomena with both spatial and temporal (spatiotemporal) evolution using a network of distributed sensors is a critical problem in many controls applications. Consider, for example, a team of robots with the task of destroying herbicide-resistant weeds on a farm (see Figure 1, also see "How can control engineers help agriculture?"). This team of robots needs to predict weed growth across the whole farm to make intelligent, coordinated decisions [1]. However, the robots can only observe a limited part of the field at a time, leading to the critical problem: How can a few robots which can only partially observe the field at any time predict the full state of the spatiotemporally-evolving weed growth over the entire field? When building an observer over a spatiotemporal process, which locations should be sampled to obtain the necessary information to render the problem observable? The goal of this tutorial is to show the steps we have taken towards addressing this kind of challenging problem. Examples of such problems abound across many domains, including: modeling and monitoring of ocean heat content and acidification for oceanography using a network of satellites and surface sensors [2]; prediction of traffic patterns using data from vehicles, cellphones, and traffic cameras; prediction of enemy movements through ground and aerial surveillance; and prediction of extreme weather events using data from weather stations and aerial drones [3]. The rapid advances in the computational power of compact systems and robotics as a whole has led to an explosion of real-world applications for such distributed cyber-physical systems.

These types of applications need to estimate complex, stochastic dynamics that are distributed over space and time. The key constraint is the number of spatially distributed sensors available, which are not enough to entirely cover the whole space at any given moment in

time. One approach to solving the problem under this constraint is to use a predictive model of the phenomenon which informs our sensing strategy. While the modeling of such spatiotemporal phenomena has traditionally been the object of study in geostatistics, it has in recent years gained more attention in the machine learning community [4]. The data-driven models developed by machine learning techniques provide a way to capture complex spatiotemporal phenomena which are not easily modeled by first-principles alone. However, these models are limited by the data sets they are trained on, and the high variability of complex distributed physical systems makes it even more challenging. For example, a model trained over years of weed growth data in one field doesn't necessarily generalize to another, and a model trained on the past few years of data still cannot reliably predict weather variability in the following year. What is needed is not just a *predict* system but a *predict-and-correct* system. This tutorial shows how to design just such a system for these kind of problems. The system utilizes kernel methods for modeling, Bayesian filtering theory for prediction correction, and exploits the mathematical structure of the regression and dynamics models to place the sensors.

In the machine learning community, kernel methods represent a class of well-studied and powerful methods for regression in spatial domains. In these techniques, correlations between the input variables are related via covariance kernels, and the model is generated by a linear combination of the kernels [5]–[7]. In recent years, kernel methods have been applied to spatiotemporal regression problems with varying degrees of success [4], [5]. Many recent methods have focused on nonstationary covariance kernel design and algorithms for learning the associated hyperparameters [8]–[11]. These methods, which focus on the careful design of covariance kernels, have been proposed as an alternative to the naive approach of simply including time as an additional input dimension in the kernel [12]. The careful design/optimization of a covariance kernel avoids an explosion in the number of parameters used by the model, which would be inevitable in the naive approach, and can better account for spatiotemporal couplings. Such covariance kernels, however, do not scale in the face of large-scale phenomena since the optimization of the kernel hyperparameters is non-convex and computationally demanding for large datasets [13]. Deep learning also suffers from similar issues, and moreover lacks the spatial encoding properties of certain kernels, which are exploited by the strategy outlined in this tutorial.

No matter how much data the model is trained on, it cannot completely capture the variability of real-world systems with complex spatiotemporal dynamics. This is a problem that the controls community is quite aware of; their solution to this is built upon feedback, leading to fundamental notions of observability and controllability which can be used to build robust state estimators and controllers. To bring these ideas to fruition in the spatiotemporal problem, we needed a way to find where sensing/control should be performed to ensure that

the state estimation problem can be made observable/controllable. While methods such as [11] have succeeded in using nonstationary kernels that evolve efficiently using feedback, it is not clear how notions of observability and controllability could be utilized with it or other existing kernel-based machine learning models, and how observers and controllers can be embedded within such models. Computational challenges can be addressed with faster methods or increasing computational power; however, addressing the latter, more fundamental challenge in designing robust observers/controllers is particularly important in the design of reliable engineering systems, such as distributed sensor/actuator networks intended for monitoring physical phenomena, autonomous soft-robots, or other physical systems with distributed sensing and actuation.

Contributions

In this tutorial, we present a different perspective on solving the spatiotemporal monitoring problem that brings together kernel-based modeling, systems theory, and Bayesian filtering. We define the monitoring problem as follows: *given an approximate predictive model of the spatiotemporal phenomena learned using historical data, estimate the current latent state of the phenomena in the presence of uncertainty, using as few sensors as possible*. Ideally, the solution to the problem should also provide guidance on how many sensors are needed and where to place them. In this paper we argue that when it comes to predictive inference over spatiotemporal phenomena, a Kalman-filter type approach of predicting and correcting with feedback from a set of minimal sensors is a robust way of dealing with real-world uncertainties and inherent modeling errors. In the context of this specific problem, our main contributions are two-fold: first, we demonstrate that spatiotemporal functional evolution can be modeled using stationary kernels with a linear dynamical systems layer on their mixing weights. In particular, in contrast with existing work, this approach does not necessarily require the design of complex spatiotemporal kernels, and can accommodate positive-definite kernels on any domain on which it is possible to define them, which includes non-Euclidean domains such as Riemannian manifolds, strings, graphs and images [14]. Second, we show that such a model can be utilized to determine sensing locations that guarantee that the hidden states of functional evolution can be estimated using a Bayesian state-estimator (Kalman filter) that is embedded in the feature space of the kernel model with very few sensors. A benefit of our solution's approach is that it provides guidance on how many sensors are needed and where to place them. Accordingly, we provide sufficient conditions on the number and location of sensor measurements required and prove non-conservative lower bounds on the minimum number of sampling locations by developing fundamental results on the observability of kernel-based models. Our model is also analyzed in terms of Koopman operator theory, and several key theoretical results are proven showing that our model can produce the Koopman modes, eigenvalues, and eigenfunctions. The validity of the

presented model and sensing techniques is corroborated using synthetic and large real datasets.

Broader Context

The fundamental idea of building observers and controllers embedded in the feature spaces of machine learning models introduced in this paper is generalizable beyond the particular application of spatiotemporal monitoring. Figure 2 shows a general landscape of problems that are relevant for engineering. Since the controls literature is strongest when the system dynamics can be represented as Ordinary Differential Equations (ODEs), some of the major successes of controls have included results such as Linear Quadratic Gaussian control, reinforcement learning, and adaptive control in state spaces with well-defined, finite, and *physically meaningful* state variables. The estimation of the states of these temporally evolving, finite-dimensional state-space systems have been extensively studied in the context of Kalman filtering and observer design [15]. A different approach for modeling complex spatiotemporal dynamical systems comes from machine learning, where trained models reside in abstract feature spaces that are only relatable to physical quantities through complex functional operations (see "Feature Spaces in Machine Learning"). There has been some work that relates approaches from controls to machine learning (see e.g. [16] for an extension of the Kalman filter to the functional domain), but the results are not studied in the context of the spatiotemporal monitoring problem presented here.

To fuse machine learning with controls for building robust systems for engineering, we need to answer fundamental questions such as 1) the least number of sensors required to observe a distributed system, 2) the placement of sensors/actuators to guarantee observability/controllability of the system, and 3) the effect of random sensor placement on system observability/controllability.

This tutorial presents an approach that can provide one formal way of addressing these and other questions about complex systems that are modeled with machine learning. In particular, we demonstrate how linear dynamical systems can be embedded in the reproducing kernel Hilbert space (RKHS) [6], [17], [18] generated by features used in Gaussian Process modeling [19], [20], and utilized to answer fundamental questions such as controllability and observability. We expect that follow-up work will exploit the framework presented in this paper of utilizing linear models in RKHSs and feature spaces of other machine learning models to enable practical and analyzable data-driven engineering systems. To facilitate the development of the theory, we have focused this paper on the problem of monitoring spatiotemporal phenomena. However, the idea should be generalizable to any distributed cyber-physical system that is changing with space and time.

Outline of the article and relationship to prior work by the authors

We begin the rest of the article by summarizing some related work in machine learning in this area. "Feature Spaces in Machine Learning" discusses the concept of feature spaces and its importance in machine learning in a broader context. We then formulate the problem, introduce *Kernel Observers (KO)*, and develop the main theoretical and algorithmic results. This is followed by some results on the expected number of randomly placed sensors required to monitor a spatiotemporal process in the context of our model. An extension to the KO method called *Evolving Gaussian Processes (E-GP)* is presented that learns one model for multiple, similar spatiotemporal processes, the efficacy of which on real-world CFD data is presented in Sidebar *Learning Fluid Flows with Evolving Gaussian Processes*. Elements of the work presented in this paper first appeared in Neural Information Processing Systems (NIPS 2016) ([21], [22]), IEEE CDC 2015 conference [23], the Conference on Robot Learning (CoRL 2017) [24] and IEEE ACC 2018 conference [25]. This paper presents a comprehensive set of results and fills in the missing links in a single encompassing publication, and introduces new results on observability in the presence of random sensor placement, and connections to Koopman operator theory. As such, we have focused in this article mostly on the fundamental theory and practical algorithms for modeling, estimation, and control, while the details of how to optimally implement the presented algorithms are omitted. Instead, an open-source code-base is made available in MATLAB at `http://daslab.illinois.edu/software.html` or at `https://github.com/hkingravi/FunctionObservers` and in Python on GitHub at `https://github.com/hkingravi/funcobspy`.

Related Work

There is a large amount of literature on Gaussian Processes and spatiotemporal modeling, a complete survey of which is beyond the scope of this paper. Since our contributions are in the area of creating a feedback-based observer in the feature spaces of GP models, we will here discuss related work in three related areas: spatiotemporal modeling with GPs, the connection of GPs to Kalman filtering, and sensor placement for inference in spatiotemporal domains.

The use of process-dependent kernels for spatiotemporal modeling in geostatistics is well-studied [4], [26], [27]. Other approaches that utilize hierarchy or evolution of kernels have also been used for modeling spatiotemporal functions [28]–[30]. From the machine learning perspective, a naive approach is to utilize both spatial and temporal variables as inputs to a Mercer kernel [31]. However, this technique leads to an ever-growing kernel dictionary. Furthermore, constraining the dictionary size or utilizing a moving window will occlude learning of long-term patterns. A more clever approach is to use state-space representations of time-varying GPs [28],

[32]. From this viewpoint, each GP instance is viewed as a snapshot of an evolving set of weights. We follow in a similar vein here, with added emphasis on exploiting mathematical structures relevant to observability and controllability. Periodic or nonstationary covariance functions and nonlinear transformations have been proposed for spatiotemporal modeling [5], [9]. Work focusing on nonseparable and nonstationary covariance kernels seeks to design kernels optimized for environment-specific dynamics, and to tune their hyperparameters in local regions of the input space. Seminal work in [33] proposes a process convolution approach for space-time modeling. This model captures nonstationary structure by allowing the convolution kernel to vary across the input space. This approach can be extended to a class of nonstationary covariance functions, thereby allowing the use of a Gaussian process (GP) framework, as shown in [34]. However, since this model's hyperparameters are inferred using MCMC integration, its application has been limited to smaller datasets. To overcome this limitation, [10] proposes to use the mean estimates of a second isotropic GP (defined over latent length scales) to parameterize the nonstationary covariances. Finally, [8] considers non-isotropic variation across different dimensions of input space for the second GP as opposed to isotropic variation by [10]. Issues with this line of approach include the nonconvexity of the hyperparameter optimization problem and the fact that selection of an appropriate nonstationary covariance function for the task at hand is a nontrivial design decision (as noted in [35]).

Apart from directly modeling the covariance function using additional latent GPs, there exist several other approaches for specifying nonstationary GP models. One approach maps the nonstationary spatial process into a latent space, in which the problem becomes approximately stationary [36]. Along similar lines, [37] extends the input space by adding latent variables, which allows the model to capture nonstationarity in original space. Both these approaches require MCMC sampling for inference, and as such are subject to the limitations mentioned in the preceding paragraph. A geostatistics approach that finds dynamical transition models on the linear combination of weights of a parameterized model [4], [16] is advantageous when the spatial and temporal dynamics are hierarchically separated, leading to a convex learning problem. This approach has been utilized previously in MRI imaging [38], [39]. As a result, complex nonstationary kernels are often not necessary (although they can be accommodated). This approach is essentially the starting point of this tutorial. A systems-theoretic study of this viewpoint enables our fundamental contributions, which are 1) allowing for inference on more general domains with a larger class of basis functions than those typically considered in the geostatistics community, and 2) quantifying the minimum number of measurements required to estimate the state of the system.

Kalman filtering in the context of Gaussian processes and kernel models has also been

quite widely studied [27], [40]–[43]. There is a direct link between the Bayesian approach to inference taken in GPs and its natural extension to Kalman Filters. Our contributions here are in creating explicit connections between feedback observers and inference by deriving conditions on observability in the kernel space. This leads to explicit conditions on the number of sensors required and where to place them. Lastly, sensor placement optimization is also a well-studied area. Examples include, but are not limited to 1) geometric approaches, which seek to provide a covering of the operating space without making assumptions about the spatiotemporal dynamics [44], and 2) information-theoretic approaches, which place their focus on sensor placement optimizing strategies based on mutual information and information entropy for Gaussian process models [45]. It should be noted that the contribution of our work concerning sensor placement is to provide *sufficient conditions* for monitoring rather than optimization of the placement locations, and therefore a comparison with these approaches is not considered in the experiments.

Lastly, we connect our work to the large body of literature produced in the last decade on Koopman operator theory and Dynamic Mode Decomposition, particularly in the Computational Fluid Dynamics (CFD) community. These methods rely on discovering *modes* of motion which show the spatial distribution, oscillation frequency, and growth rate/decay of the component dynamics of the system. Many applications have been realized through these methods, which include the ability to transform the state space so the dynamics appear linear, to predict the temporal evolution of the linear system, to reconstruct the state of the original nonlinear system, and even to implement controller design. Dynamic Mode Decomposition (DMD) is the most widely used method for finding a finite-dimensional subspace of the Koopman operator's infinite-dimensional domain to work in [46]. Williams et al., recently integrated DMD with the kernel trick, allowing the algorithm to be extended to systems with much larger dimensions [47]. Brunton et al., inspired by DMD, were able to generate governing equations from data by sparse identification of nonlinear dynamical systems [48]. However, these methods are restricted to approximating the Koopman operator given a fixed vector-valued observable, and have no way of effectively using measurements that vary in both number and location over time. Furthermore, the state of research into data-driven generalizing over similar systems with varying parameters is at best preliminary.

Kernel Observers

This section outlines our modeling framework and presents theoretical results associated with the number of sampling locations required for monitoring functional evolution.

Problem Formulation

We focus on predictive inference of a time-varying stochastic process, whose mean f evolves temporally via $f_{\tau+1} \sim \mathbb{F}(f_\tau, \eta_\tau)$, where $\mathbb{F}$ is a distribution varying with time τ and exogenous inputs η. Our approach builds on the fact that in several cases, temporal evolution can be hierarchically separated from spatial functional evolution. A classical and quite general example of this is the *abstract evolution equation* (AEO), which can be defined as the evolution of a function u embedded in a Banach space $\mathcal{B}$: $\dot{u}(t) = \mathcal{L}u(t)$, subject to $u(0) = u_0$, and $\mathcal{L} : \mathcal{B} \to \mathcal{B}$ determines spatiotemporal transitions of $u \in \mathcal{B}$ [49]. This model of spatiotemporal evolution is very general (AEOs, for example, model many PDEs), but working in Banach spaces can be computationally taxing. A simple way to make the approach computationally feasible is to place restrictions on $\mathcal{B}$: in particular, we restrict the sequence f_τ to lie in a Reproducing Kernel Hilbert Space (RKHS), the theory of which provides powerful tools for generating flexible classes of functions with relative ease [5]. In a kernel-based model, $k : \Omega \times \Omega \to \mathbb{R}$ is a positive-definite Mercer kernel on a domain Ω that models the covariance between any two points in the input space, and implies the existence of a smooth map $\psi : \Omega \to \mathcal{H}$, where $\mathcal{H}$ is an RKHS with the property $k(x, y) = \langle \psi(x), \psi(y) \rangle_{\mathcal{H}}$. The key insight behind the proposed model is that spatiotemporal evolution in the input domain corresponds to temporal evolution of the mixing weights of a kernel model alone in the functional domain. Therefore, f_τ can be modeled by tracing the evolution of its mean embedded in a RKHS using switched ordinary differential equations (ODE) when the evolution is continuous, or switched difference equations when it is discrete (Figure 3). The advantage of this approach is that it allows us to utilize powerful ideas from systems theory for deriving necessary and sufficient conditions for spatiotemporal monitoring.

In this paper, we restrict our attention to the class of functional evolutions $\mathbb{F}$ defined by linear Markovian transitions in an RKHS. While extension to the nonlinear case is possible (and non-trivial), it is not pursued in this paper to help ease the exposition of the key ideas. The class of linear transitions in RKHS is rich enough to approximately model many real-world datasets, as suggested by our experiments.

Let $y \in \mathbb{R}^N$ be the measurements of the function available from N sensors, $\mathcal{A} : \mathcal{H} \to \mathcal{H}$ be a linear transition operator in the RKHS $\mathcal{H}$, and $\mathcal{K} : \mathcal{H} \to \mathbb{R}^N$ be a linear measurement operator. The model for the functional evolution and measurement studied in this paper is:

$$f_{\tau+1} = \mathcal{A} f_\tau + \eta_\tau, \quad y_\tau = \mathcal{K}_\tau f_\tau + \zeta_\tau, \tag{1}$$

where η_τ is a zero-mean stochastic process in $\mathcal{H}$, and ζ_τ is a Wiener process in $\mathbb{R}^N$. Classical treatments of kernel methods emphasize that for most kernels, the feature map ψ is unknown,

and possibly infinite-dimensional; this forces practioners to work in the dual space of $\mathcal{H}$, whose dimensionality is the number of samples in the dataset being modeled. This conventional wisdom precludes the use of kernel methods for most tasks involving modern datasets, which may have millions and sometimes billions of samples [50]. An alternative is to work with a feature map $\widehat{\psi}(x) := \left[\, \widehat{\psi}_1(x) \;\cdots\; \widehat{\psi}_M(x) \,\right]^T$ to an approximate feature space $\widehat{\mathcal{H}}$ with the property that for every element $f \in \mathcal{H}$, $\exists \widehat{f} \in \widehat{\mathcal{H}}$ s.t. $\|f - \widehat{f}\| < \epsilon$ for an appropriate function norm and $\epsilon > 0$. A few such approximations are listed below.

Dictionary of atoms: Let Ω be compact. Given points $\mathcal{C} = \{c_1, \ldots, c_M\}$, $c_i \in \Omega$, we have a dictionary of atoms $\mathcal{F}^{\mathcal{C}} = \{\psi(c_1), \cdots, \psi(c_M)\}$, $\psi(c_i) \in \mathcal{H}$, the span of which is a strict subspace $\widehat{\mathcal{H}}$ of the RKHS $\mathcal{H}$ generated by the kernel. Here,

$$\widehat{\psi}_i(x) := \langle \psi(x), \psi(c_i) \rangle_{\mathcal{H}} = k(x, c_i). \tag{2}$$

Low-rank approximations: Let Ω be compact, let $\mathcal{C} = \{c_1, \ldots, c_M\}$, $c_i \in \Omega$, and let $K \in \mathbb{R}^{M \times M}$, $K_{ij} := k(c_i, c_j)$ be the Gram matrix computed from $\mathcal{C}$. This matrix can be diagonalized to compute approximations $(\widehat{\lambda}_i, \widehat{\phi}_i(x))$ of the eigenvalues and eigenfunctions $(\lambda_i, \phi_i(x))$ of the kernel [51]. These spectral quantities can then be used to compute $\widehat{\psi}_i(x) := \sqrt{\widehat{\lambda}_i}\widehat{\phi}_i(x)$.

Random Fourier features: Let $\Omega \subset \mathbb{R}^n$ be compact, and let $k(x, y) = e^{-\|x-y\|^2/2\sigma^2}$ be the Gaussian RBF kernel. Then random Fourier features approximate the kernel feature map as $\widehat{\psi}_\omega : \Omega \to \widehat{\mathcal{H}}$, where ω is a sample from the Fourier transform of $k(x, y)$, with the property that $k(x, y) = \mathbb{E}_\omega[\langle \widehat{\psi}_\omega(x), \widehat{\psi}_\omega(y) \rangle_{\widehat{\mathcal{H}}}]$ [50]. In this case, if $V \in \mathbb{R}^{M/2 \times n}$ is a random matrix representing the sample ω, then $\widehat{\psi}_i(x) := \left[\, \frac{1}{\sqrt{M}} \sin([Vx]_i), \frac{1}{\sqrt{M}} \cos([Vx]_i) \,\right]$. Similar approximations exist for other radially symmetric kernels, as well as dot-product kernels.

In the approximate space case, we replace the transition operator $\mathcal{A} : \mathcal{H} \to \mathcal{H}$ in (1) by $\widehat{\mathcal{A}} : \widehat{\mathcal{H}} \to \widehat{\mathcal{H}}$. This approximate regime, which combines the flexibility of a truly nonparametric approach with computational realizability, still allows for the representation of rich phenomena, as will be seen in the sequel, and in Figure 5. The finite-dimensional evolution equations approximating (1) in dual form are:

$$w_{\tau+1} = \widehat{A} w_\tau + \eta_\tau, \quad y_\tau = K w_\tau + \zeta_\tau, \tag{3}$$

where we have matrices $\widehat{A} \in \mathbb{R}^{M \times M}$, $K \in \mathbb{R}^{N \times M}$, the vectors $w_\tau \in \mathbb{R}^M$, and where we have slightly abused notation to let y_τ, η_τ and ζ_τ denote their $\widehat{\mathcal{H}}$ counterparts. Here K is the matrix whose rows are of the form $K_{(i)} = \widehat{\Psi}(x_i) = \left[\, \widehat{\psi}_1(x_i) \;\; \widehat{\psi}_2(x_i) \;\cdots\; \widehat{\psi}_M(x_i) \,\right]$. In systems-theoretic language, each row of K corresponds to a *measurement* at a particular location, and the matrix itself acts as a measurement operator.

The equations in (1) suggest an immediate extension to functional control problems. Pick another basis for $\mathcal{H}$ as $\widetilde{\psi}(x) := \begin{bmatrix}\widetilde{\psi}_1(x) & \cdots & \widetilde{\psi}_{\ell'}(x)\end{bmatrix}^T$, where the functions $\widetilde{\psi}_j(x)$ are used to approximate the RKHS $\mathcal{H}$ generated by the kernel. We denote the span of these functions as $\widetilde{\mathcal{H}}$. In the dictionary of atoms case, an example would be another set of atoms $\mathcal{F}_D = \begin{bmatrix}\psi(d_1) & \cdots & \psi(d_{\ell'})\end{bmatrix}$, $\psi(d_j) \in \mathcal{H}$, $d_j \in \Omega$, with $\widetilde{\mathcal{H}}$ being a strict subspace of the RKHS $\mathcal{H}$ generated by the kernel. The functional evolution equation is then as follows:

$$f_{\tau+1} = \mathcal{A}f_\tau + \mathcal{B}\delta_\tau + \eta_\tau, \quad y_\tau = \mathcal{K}_\tau f_\tau + \zeta_\tau, \tag{4}$$

where the control functions δ_τ evolve in $\widetilde{\mathcal{H}}$, and $\mathcal{B} : \widetilde{\mathcal{H}} \to \widehat{\mathcal{H}}$. To derive the finite-dimensional equivalent of $\mathcal{B}$, we have to work out the structure of the matrix B: since $\widehat{\mathcal{H}}$ is not, in general, isomorphic to $\widetilde{\mathcal{H}}$, this imposes strict restrictions on B. We can derive B using least squares using the inner product of $\mathcal{H}$. An instructive example is where both $\widehat{\mathcal{H}}$ and $\widetilde{\mathcal{H}}$ are generated by dictionaries of atoms; recall that in this case, $\mathcal{F}^{\mathcal{C}} = \begin{bmatrix}\psi(c_1) & \cdots & \psi(c_M)\end{bmatrix}$ is the basis for $\widehat{\mathcal{H}}$, and let $\delta = \sum_{j=1}^{\ell'} \acute{w}_j \psi(d_j)$, and let $\mathcal{F}^{\mathcal{C}} = \begin{bmatrix}\psi(c_1) & \cdots & \psi(c_M)\end{bmatrix}$ be the basis for $\mathcal{H}^{\mathcal{C}}$. Then the projection of δ onto $\widehat{\mathcal{H}}$ can be derived as:

$$\begin{bmatrix}\langle \delta, \psi(c_1)\rangle_{\mathcal{H}} \\ \vdots \\ \langle \delta, \psi(c_M)\rangle_{\mathcal{H}}\end{bmatrix} = \underbrace{\begin{bmatrix}\langle \psi(d_1), \psi(c_1)\rangle_{\mathcal{H}} & \cdots & \langle \psi(d_{\ell'}), \psi(c_1)\rangle_{\mathcal{H}} \\ \vdots & \ddots & \vdots \\ \langle \psi(d_1), \psi(c_M)\rangle_{\mathcal{H}} & \cdots & \langle \psi(d_{\ell'}), \psi(c_M)\rangle_{\mathcal{H}}\end{bmatrix}}_{K_{CD}} \begin{bmatrix}\acute{w}_1 \\ \vdots \\ \acute{w}_{\ell'}\end{bmatrix}. \tag{5}$$

Note that in the dictionary of atoms case, the entries of K_{CD} can be computed in closed form as $K_{CDij} := k(d_i, c_j)$, using the reproducing property. This derivation shows that the operator B is simply $K_{CD} \in \mathbb{R}^{M \times \ell'}$, the kernel matrix between the data C generating the atoms $\mathcal{F}^{\mathcal{C}}$ of $\widehat{\mathcal{H}}$ and the data D generating the atoms $\mathcal{F}_D$ of $\widetilde{\mathcal{H}}$.

Thus, the finite-dimensional evolution equations equivalent to (4) are

$$w_{\tau+1} = \widehat{A}w_\tau + K_{CD}\acute{w}_\tau, \quad y_\tau = K_\tau w_\tau. \tag{6}$$

We define the *generalized observability matrix* [52] as $\mathcal{O}_\Upsilon = \begin{bmatrix} K\widehat{A}^{\tau_1} \\ \cdots \\ K\widehat{A}^{\tau_L}\end{bmatrix}$ where $\Upsilon = \{\tau_1, \ldots, \tau_L\}$ are the set of instances τ_i when we apply the operator K. A linear system is said to be *observable* if $\mathcal{O}_\Upsilon$ has full column rank (that is, $\mathrm{Rank}(\mathcal{O}_\Upsilon) = M$) for $\Upsilon = \{0, 1, \ldots, M-1\}$ [52]. Observability guarantees two critical facts: firstly, it guarantees that the state w_0 can be recovered exactly from a finite series of measurements $\{y_{\tau_1}, y_{\tau_2}, \ldots, y_{\tau_L}\}$; in particular, defining $y_\Upsilon = \begin{bmatrix}y_{\tau_1}^T, y_{\tau_2}^T, \cdots, y_{\tau_L^T}\end{bmatrix}^T$, we have that $y_\Upsilon = \mathcal{O}_\Upsilon w_0$. Secondly, it guarantees that a feedback based *observer* can be designed such that the estimate of w_τ, denoted by $\widehat{w}_\tau$, converges exponentially fast to w_τ in the limit of

samples. Note that all our theoretical results assume $\widehat{A}$ is available: while we perform system identification in the experiments ([21]), it is not the focus of the paper.

We are now in a position to formally state the spatiotemporal modeling, control, and inference problems being considered: given a spatiotemporally evolving system modeled using (3), choose a set of N sensing locations such that even with $N \ll M$, the functional evolution of the spatiotemporal model can be estimated (which corresponds to *monitoring*), can be predicted robustly (which corresponds to *Bayesian filtering*), and which can be controlled (which corresponds to *functional control*). Our approach to solve the monitoring and prediction problem relies on the design of the measurement operator K so that the pair $(K, \widehat{A})$ is observable: any Bayesian state estimator (for example a Kalman filter) utilizing this pair is denoted as a **kernel observer**. In the case where no measurements are taken, for the sake of consistency, we denote the state estimator as an *autonomous* kernel observer. In the controls case, given a spatiotemporally evolving system modeled using (6), we need to choose a set of N sensing locations and ℓ' control locations, such that even with $N \ll M,\ \ell' \ll M$, the functional evolution of the spatiotemporal model can be controlled; in this case, we must design both a measurement operator K and a control operator K_{CD} such that the pair $(K_{CD}, \widehat{A})$ is controllable: a controls system utilizing this pair and the measurement operator K is denoted as a **kernel controller**.

Preliminaries on Rational Canonical Structures

We take a geometric approach towards the choice of sampling locations for inferring w_τ in (3); the extension for control is similar. We use the notation $\mathcal{V}$, with $\dim(\mathcal{V}) = M$, to emphasize the fact that these theorems hold for any finite-dimensional vector space. Consider the linear operator $\mathcal{A} : \mathcal{V} \to \mathcal{V}$, and recall that the definition of observability requires the construction of a linear operator $\mathcal{K} : \mathcal{V} \to \mathcal{U}$, with $\dim(\mathcal{U}) = N$, such that $\operatorname{rank}\left[(\mathcal{K})^T \cdots \left(\mathcal{K}\mathcal{A}^{M-1}\right)^T \right]^T = M$. In most applications, if $N \geq M$, and $\operatorname{rank}(\mathcal{K}) = N$, it is reasonable to expect that observability may be achieved. However, for our purposes, N must be *significantly* less than M. Therefore, we must design $\mathcal{K}$ with as small a rank as possible. To do so, we require a series of vectors v_i that, under repeated iterations of $\mathcal{A}$, can generate a basis for $\mathcal{V}$. For this task, we will use a fundamental decomposition result from the theory of modules, known as the *rational canonical structure* of $\mathcal{A}$ [53]. The intuition here is that if the sequence $\{v_i\}_i$ can generate this basis, it can be directly used to construct $\mathcal{K}$.

The linear operator $\mathcal{A} : \mathcal{V} \to \mathcal{V}$ has a characteristic polynomial $\pi(\lambda)$ such that $\pi(\mathcal{A}) = 0$ by the Cayley-Hamilton theorem. The minimal polynomial (MP) of $\mathcal{A}$ is the monic polynomial $\alpha(\cdot)$ of least degree (denoted by $\deg(\cdot)$) given as $\alpha(\lambda) = a_0 + a_1\lambda + \cdots + \lambda^{\deg(\alpha)} = 0$, such that $\alpha(\mathcal{A}) = a_0 I + a_1\mathcal{A} + \cdots + \mathcal{A}^{\deg(\alpha)} = \mathbf{0}$. The MP is unique and divides $\pi(\lambda)$, so that

$\deg(\alpha) \leq \deg(\pi)$. The MP of a vector $v \in \mathcal{V}$ *relative to* $\mathcal{A}$ is the unique monic polynomial ξ_v of least degree such that $\xi_v(\mathcal{A})v = a_0 v + a_1 \mathcal{A}v + \cdots + \mathcal{A}^{\deg(\alpha)}v = 0$. If $\deg(\alpha) = M$, then $\mathcal{A}$ is *cyclic* and $\exists v \in \mathcal{V}$, such that the vectors $\{v, \mathcal{A}v, \ldots, \mathcal{A}^{M-1}v\}$ form a basis for $\mathcal{V}$; this is the same as saying that the pair $(v^T, \mathcal{A}^T)$ is observable. A subspace $\mathcal{V}_\mathcal{S} \subset \mathcal{V}$ s.t. $\mathcal{A}\mathcal{V}_\mathcal{S} \subset \mathcal{V}_\mathcal{S}$ is $\mathcal{A}$-*cyclic* if $\mathcal{A}_{|\mathcal{V}_\mathcal{S}}$, the restriction of $\mathcal{A}$ to the subspace $\mathcal{V}_\mathcal{S}$, is cyclic. If $\alpha(\lambda)$ is the minimal polynomial of $\mathcal{A}$ and $\deg(\alpha) = m < M$, $\exists\ v \in \mathcal{V}$ such that $\{v, \mathcal{A}v, \ldots, \mathcal{A}^{m-1}v\}$ span an m-dimensional $\mathcal{A}$-cyclic subspace $\mathcal{V}_\mathcal{S}$, with v being the *cyclic generator* of $\mathcal{V}_\mathcal{S}$. The subspace $\mathcal{V}_\mathcal{S}$ decomposes $\mathcal{V}$ relative to $\mathcal{A}$. By the rational canonical structure theorem (Theorem 0.1 of [53]), $\mathcal{A}$ can be successively decomposed into subspaces $\mathcal{V}_i \subset \mathcal{V}$, $i \in \{1, \ldots, \ell\}$, s.t. $\mathcal{V} = \mathcal{V}_1 \oplus \ldots \oplus \mathcal{V}_\ell$, $\mathcal{A}\mathcal{V}_i \subset \mathcal{V}_i$, and $\mathcal{A}_{|\mathcal{V}_i}, i \in \{1, \ldots, \ell\}$, are cyclic. In general, the subspaces $\mathcal{V}_i$ are not unique for a fixed $\mathcal{A}$. The integer ℓ is unique and is called the *cyclic index of* $\mathcal{A}$.

One of our main results is to show that the cyclic index is a lower bound on the number of measurements required to reconstruct w_τ (see Prop. 3 and Alg. 1 below). The matrix transform associated to this theorem is known as the *Frobenius normal form* (denoted by $C \in \mathbb{R}^{M\times M}$): for $\mathcal{A} \in \mathbb{R}^{M\times M}$, $\exists Q \in \mathbb{R}^{M\times M}$ invertible such that $\mathcal{A} = QCQ^{-1}$. We will also use the *Jordan decomposition*, where for $\mathcal{A} \in \mathbb{R}^{M\times M}$, $\exists P \in \mathbb{R}^{M\times M}$ invertible such that $\mathcal{A} = P\Lambda P^{-1}$, where Λ is a unique block diagonal matrix with Jordan blocks with λ_i along the diagonal. If all the eigenvalues λ_i are nonzero and real, we say the matrix has a *full-rank Jordan decomposition*.

Main Results

In this section, we prove results concerning the observability of spatiotemporally varying functions modeled by the functional evolution and measurement equations (3). In particular, observability of the system states implies that we can recover the current state of the spatiotemporally varying function using a small number of sampling locations N, which allows us to 1) track the function, and 2) predict its evolution forward in time. We work with the approximation $\widehat{\mathcal{H}} \approx \mathcal{H}$: given M basis functions, this implies that the dual space of $\widehat{\mathcal{H}}$ is $\mathbb{R}^M$. Proposition 1 shows that if $\widehat{A}$ has a full-rank Jordan decomposition, the observation matrix K meeting a condition called *shadedness* (Definition 1) is sufficient for the system to be observable. Proposition 2 provides a lower bound on the number of sampling locations required for observability which holds for any $\widehat{A}$. Proposition 3 constructively shows the existence of an abstract measurement map $\widetilde{K}$ achieving this lower bound. Since the measurement map does not have the structure of a kernel matrix, a slightly weaker sufficient condition for the observability of any $\widehat{A}$ is in Theorem 1. Finally, since both K and K_{CD} are kernel matrices generated from a shared kernel, these observability results translate directly into controllability results.

Definition 1: **(Shaded Observation Matrix)** Given $k : \Omega \times \Omega \to \mathbb{R}$ positive-definite on

a domain Ω, let $\{\widehat{\psi}_1(x), \ldots, \widehat{\psi}_M(x)\}$ be the set of bases generating an approximate feature map $\widehat{\psi} : \Omega \to \widehat{\mathcal{H}}$, and let $\mathcal{X} = \{x_1, \ldots, x_N\}$ be the set of sampling (or sensing) locations, with each $x_i \in \Omega$. Let $K \in \mathbb{R}^{N \times M}$ be the observation matrix, where $K_{ij} := \widehat{\psi}_j(x_i)$. For each row $K_{(i)} := [\, \widehat{\psi}_1(x_i) \cdots \widehat{\psi}_M(x_i) \,]$, define the set $\mathcal{I}_{(i)} := \{\iota_1^{(i)}, \iota_2^{(i)}, \ldots, \iota_{M_i}^{(i)}\}$ to be the indices in the observation matrix row i which are nonzero. Then if

$$\bigcup_{i \in \{1,\ldots,N\}} \mathcal{I}^{(i)} = \{1, 2, \ldots, M\},$$

we denote K as a *shaded observation matrix* (see Figure 4(a)).

This definition seems quite abstract, so the following remark considers a more concrete example.

Remark 1: Let $\widehat{\psi}$ be generated by the dictionary given by $\mathcal{C} = \{c_1, \ldots, c_M\}$, $c_i \in \Omega$. Note that since $\widehat{\psi}_j(x_i) = \langle \psi(x_i), \psi(c_j) \rangle_{\mathcal{H}} = k(x_i, c_j)$, K is the kernel matrix between $\mathcal{X}$ and $\mathcal{C}$. For the kernel matrix to be shaded thus implies that there does not exist an atom $\psi(c_j)$ such that the projections $\langle \psi(x_i), \psi(c_j) \rangle_{\mathcal{H}}$ vanish for all x_i, $1 \leq i \leq N$. Intuitively, the shadedness property requires that the sensor locations x_i are privy to information propagating from every c_j. As an example, note that, in principle, for the Gaussian kernel, a single row generates a shaded kernel matrix. However, in this case, the matrix can have many entries that are extremely close to zero, and will probably be very ill-conditioned.

Proposition 1: Given $k : \Omega \times \Omega \to \mathbb{R}$ positive-definite on a domain Ω, let $\{\widehat{\psi}_1(x), \ldots, \widehat{\psi}_M(x)\}$ be the set of bases generating an approximate feature map $\widehat{\psi} : \Omega \to \widehat{\mathcal{H}}$, and let $\mathcal{X} = \{x_1, \ldots, x_N\}$, $x_i \in \Omega$. Consider the discrete linear system on $\widehat{\mathcal{H}}$ given by the evolution and measurement equations (3). Suppose that a full-rank Jordan decomposition of $\widehat{A} \in \mathbb{R}^{M \times M}$ of the form $\widehat{A} = P \Lambda P^{-1}$ exists, where $\Lambda = [\, \Lambda_1 \cdots \Lambda_O \,]$, and there are no repeated eigenvalues. Then, given a set of time instances $\Upsilon = \{\tau_1, \tau_2, \ldots, \tau_L\}$, and a set of sampling locations $\mathcal{X} = \{x_1, \ldots, x_N\}$, the system (3) is observable if the observation matrix K_{ij} is shaded according to Definition 1, Υ has distinct values, and $|\Upsilon| \geq M$.

Proof: To begin, consider a system where $\widehat{A} = \Lambda$, with Jordan blocks $\{\Lambda_1, \Lambda_2, \ldots, \Lambda_O\}$ along the diagonal. Then $\widehat{A}^{\tau_i} = \mathrm{diag}(\begin{bmatrix} \Lambda_1^{\tau_i} & \Lambda_2^{\tau_i} & \cdots & \Lambda_O^{\tau_i} \end{bmatrix})$. We have that:

$$\mathcal{O}_\Upsilon = \underbrace{\begin{bmatrix} K\widehat{A}^{\tau_1} \\ \cdots \\ K\widehat{A}^{\tau_L}. \end{bmatrix}}_{\mathcal{O}_\Upsilon \in \mathbb{R}^{NL \times M}}$$

We need to prove that the column rank of $\mathcal{O}_\Upsilon$ is M, which is not immediately obvious since

typically $N \ll M$. To prove the statement, we will show that computing the rank of $\mathcal{O}_\Upsilon$ is equivalent to the rank computation of the product of two simple matrices. In what follows, we use the notation $\mathbf{0}_{\mathbb{R}^{I\times J}}$ to denote an $I \times J$ matrix of all zeros.

In the first step, we write the above matrix as the product of two matrices. Then it can be shown that $\mathcal{O}_\Upsilon$ is the product of two block matrices.

$$\mathcal{O}_\Upsilon = \underbrace{\begin{bmatrix} K & \cdots & \mathbf{0}_{\mathbb{R}^{N\times M}} \\ \vdots & \ddots & \vdots \\ \mathbf{0}_{\mathbb{R}^{N\times M}} & \cdots & K \end{bmatrix}}_{\widehat{\mathbf{K}}\in\mathbb{R}^{NL\times ML}} \underbrace{\left[\begin{array}{ccc} \Lambda_1^{\tau_1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \Lambda_O^{\tau_1} \\ \hline \vdots & \ddots & \vdots \\ \hline \Lambda_1^{\tau_L} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \Lambda_O^{\tau_L} \end{array}\right]}_{\widehat{\mathbf{A}}\in\mathbb{R}^{ML\times M}}.$$

We need to simplify $\widehat{\mathbf{K}}$ even further. Recall that a matrix's rank is preserved under a product with an invertible matrix. Design a matrix of elementary row operations $U \in \mathbb{R}^{N\times N}$ such that $\breve{K} := UK$ is a matrix with at least one row vector of nonzeros; this can be achieved by having an elementary matrix that adds rows together. By the shadedness assumption, such a matrix exists. We can write this operation as:

$$UK = \begin{bmatrix} \breve{K}_{11} & \breve{K}_{12} & \cdots & \breve{K}_{1M} \\ \breve{K}_{21} & \breve{K}_{22} & \cdots & \breve{K}_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ \breve{K}_{N1} & \breve{K}_{N2} & \cdots & \breve{K}_{NM}. \end{bmatrix}$$

Without loss of generality, and abusing notation slightly, let this multiplication lead to one nonzero row, with the rest of the elements of the matrix being zero, as:

$$UK = \begin{bmatrix} k_{11} & k_{12} & \cdots & k_{1M} \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}.$$

Since elementary matrices are full-rank, we then have that $\operatorname{rank}(UK) = \operatorname{rank}(K)$.

To analyze the rank of $\mathcal{O}_\Upsilon$, we apply these elementary matrices to every $K \in \widehat{\mathbf{K}}$. To do so, consider the block-diagonal matrix $\mathcal{U} \in \mathbb{R}^{NL\times NL}$ with $U \in \mathbb{R}^{N\times N}$ along the diagonal, and

zeros everywhere else, that is,

$$\mathcal{U} := \begin{bmatrix} U & \mathbf{0}_{\mathbb{R}^{N\times N}} & \cdots & \mathbf{0}_{\mathbb{R}^{N\times N}} \\ \mathbf{0}_{\mathbb{R}^{N\times N}} & U & \cdots & \mathbf{0}_{\mathbb{R}^{N\times N}} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}_{\mathbb{R}^{N\times N}} & \mathbf{0}_{\mathbb{R}^{N\times N}} & \cdots & U \end{bmatrix}. \tag{7}$$

It can be shown that $\mathcal{U}$ is full-rank, that is, has rank NL. Going back to the observability matrix, we have that:

$$\begin{aligned} \mathcal{U}\mathcal{O}_\Upsilon &= \mathcal{U}\widehat{\mathbf{K}}\widehat{\mathbf{A}} \\ &= \underbrace{\begin{bmatrix} UK & \mathbf{0}_{\mathbb{R}^{N\times M}} & \cdots & \mathbf{0}_{\mathbb{R}^{N\times M}} \\ \mathbf{0}_{\mathbb{R}^{N\times M}} & UK & \cdots & \mathbf{0}_{\mathbb{R}^{N\times M}} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}_{\mathbb{R}^{N\times M}} & \mathbf{0}_{\mathbb{R}^{N\times M}} & \cdots & UK \end{bmatrix}}_{\mathcal{U}\widehat{\mathbf{K}}\in\mathbb{R}^{NL\times ML}} \underbrace{\widehat{\mathbf{A}}}_{\in\mathbb{R}^{ML\times M}}, \end{aligned}$$

since $\mathbf{0}_{\mathbb{R}^{N\times N}}\mathbf{0}_{\mathbb{R}^{N\times M}} = \mathbf{0}_{\mathbb{R}^{N\times M}}$. Due to the fact that $\mathrm{rank}(\mathcal{U}\mathcal{O}_\Upsilon) = \mathrm{rank}(\mathcal{O}_\Upsilon)$, we can therefore perform our rank analysis on the simpler matrix $\mathrm{rank}(\mathcal{U}\mathcal{O}_\Upsilon)$. Note that:

$$\begin{aligned} UK\widehat{A}^{\tau_j} &= \begin{bmatrix} k_{11} & k_{12} & \cdots & k_{1M} \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} \widehat{A}^{\tau_j} \\ &= \begin{bmatrix} k_{11}\lambda_1^{\tau_j} & \binom{\tau_j}{1}\lambda_1^{\tau_j-1} + k_{12}\lambda_1^{\tau_j} & \cdots & k_{1M}\lambda_O^{\tau_j} \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & 0 \end{bmatrix}. \end{aligned}$$

Therefore, following some more elementary row operations encoded by $V \in \mathbb{R}^{ML\times ML}$, we have that:

$$\begin{aligned} V\mathcal{U}\mathcal{O}_\Upsilon &= \begin{bmatrix} k_{11}\lambda_1^{\tau_1} & \cdots & k_{1M}\lambda_O^{\tau_1} \\ k_{11}\lambda_1^{\tau_2} & \cdots & k_{1M}\lambda_O^{\tau_2} \\ \vdots & \ddots & 0 \\ k_{11}\lambda_1^{\tau_L} & \cdots & k_{1M}\lambda_O^{\tau_L} \\ \mathbf{0}_{\mathbb{R}^{M(L-1)\times 1}} & \cdots & \mathbf{0}_{\mathbb{R}^{M(L-1)\times 1}} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{\Phi} \\ \mathbf{0}_{\mathbb{R}^{M(L-1)\times M}} \end{bmatrix}. \end{aligned}$$

If the individual entries k_{1i} are nonzero, and the Jordan block diagonals have nonzero eigenvalues, the columns of $\mathbf{\Phi}$ become linearly independent. Therefore, if $L \geq M$, the column rank of $\mathcal{O}_\Upsilon$ is M, which results in an observable system.

To extend this proof to matrices $\widehat{A} = P\Lambda P^{-1}$, note that:

$$\begin{aligned}\mathcal{O}_\Upsilon &= \begin{bmatrix} K\widehat{A}^{\tau_1} \\ \cdots \\ K\widehat{A}^{\tau_L} \end{bmatrix} \\ &= \begin{bmatrix} KP\Lambda^{\tau_1}P^{-1} \\ \cdots \\ KP\Lambda^{\tau_L}P^{-1}. \end{bmatrix} \\ &= \widehat{\mathbf{K}}\boldsymbol{P}\boldsymbol{\Lambda}^t\boldsymbol{P}^{-1},\end{aligned}$$

where $\boldsymbol{P} \in \mathbb{R}^{ML\times ML}$, $\boldsymbol{\Lambda}^t \in \mathbb{R}^{ML\times ML}$, and $\boldsymbol{P^{-1}} \in \mathbb{R}^{ML\times ML}$ are the block diagonal matrices associated with the system. Since $\boldsymbol{P}$ is an invertible matrix, the conclusions about the column rank drawn before still hold, and the system is observable. ■

When the eigenvalues of the system matrix are repeated, it is not enough for K to be shaded. In the next proposition, we take a geometric approach and utilize the rational canonical form of $\widehat{A}$ to obtain a lower bound on the number of sampling locations required. Let r be the number of unique eigenvalues of $\widehat{A}$, and let γ_{λ_i} denote the geometric multiplicity of eigenvalue λ_i. Then the *cyclic index* of $\widehat{A}$ is defined as $\ell = \max_{1\leq i\leq r} \gamma_{\lambda_i}$ [53].

Proposition 2: Suppose that the conditions in Proposition 1 hold, with the relaxation that the Jordan blocks $\begin{bmatrix} \Lambda_1 & \cdots & \Lambda_O \end{bmatrix}$ may have repeated eigenvalues (that is, $\exists \Lambda_i$ and Λ_j s.t. $\lambda_i = \lambda_j$). Then there exist kernels $k(x, y)$ such that the lower bound ℓ on the number of sampling locations N is given by the cyclic index of $\widehat{A}$. In other words, the system in (3) is observable if $N \geq \ell$.

Proof: **By Contrapositive.** We will show that if the number of sampling locations are $N = \ell - 1$ (that is, $N < \ell$), then the system is not observable. Pick the Gaussian kernel in the dictionary of atoms framework, with sampling locations $x_i \in \mathcal{X}$ and centers $c_j \in \mathcal{C}$, with the additional property that $x_i \neq x_j \forall i, j \in \{1, \ldots, N\}, i \neq j$. In this case, $\mathbf{K}$ has $\ell - 1$ nonzero, linearly independent rows, and can be written as:

$$\mathbf{K} = \begin{bmatrix} k_{11} & k_{12} & \cdots & k_{1M} \\ \vdots & \vdots & \cdots & \vdots \\ k_{(\ell-1)1} & k_{(\ell-1)2} & \cdots & k_{(\ell-1)M} \end{bmatrix}.$$

Since the cyclic index is ℓ, this implies that at least one eigenvalue, say λ, has ℓ Jordan blocks. Define indices $j_1, j_2, \ldots, j_\ell \in \{1, 2, \ldots, M\}$ as the columns corresponding to the leading entries

of the ℓ Jordan blocks corresponding to λ. WLOG, let $j_1 = 1$. Using ideas similar to the last proof, we can write the observability matrix as:

$$\mathcal{O}_\Upsilon := \begin{bmatrix} k_{11}\lambda^{\tau_1} & \cdots & k_{1j_\ell}\lambda^{\tau_1} & \cdots \\ \vdots & \ddots & \vdots & \ddots \\ k_{11}\lambda^{\tau_L} & k_{1j_\ell}\lambda^{\tau_L} & \cdots & \\ \vdots & \ddots & \vdots & \ddots \\ k_{(\ell-1)1}\lambda^{\tau_1} \cdots & k_{(\ell-1)j_\ell}\lambda^{\tau_1} & \cdots & \\ \vdots & \ddots & \vdots & \ddots \\ k_{(\ell-1)1}\lambda^{\tau_L} & \cdots & k_{(\ell-1)j_\ell}\lambda^{\tau_L} & \cdots \end{bmatrix}.$$

Define $\boldsymbol{\lambda} := \begin{bmatrix} \lambda^{\tau_1} & \lambda^{\tau_2} & \cdots \lambda^{\tau_L} \end{bmatrix}^T$. Then the above matrix becomes:

$$\mathcal{O}_\Upsilon := \begin{bmatrix} k_{11}\boldsymbol{\lambda} & \cdots & k_{1j_2}\boldsymbol{\lambda} & \cdots & k_{1j_\ell}\boldsymbol{\lambda} & \cdots \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots \\ k_{(\ell-1)1}\boldsymbol{\lambda} & \cdots & k_{(\ell-1)j_2}\boldsymbol{\lambda} & \cdots & k_{(\ell-1)j_\ell}\boldsymbol{\lambda} & \cdots \end{bmatrix}.$$

We need to show that one of the columns above can be written in terms of the others. This is equivalent to solving the linear system.

$$\begin{bmatrix} k_{1j_1} \\ k_{2j_1} \\ \vdots \\ k_{(\ell-1)j_1} \end{bmatrix} = \begin{bmatrix} k_{1j_2} & \cdots & k_{1j_\ell} \\ k_{2j_2} & \cdots & k_{2j_\ell} \\ \vdots & \ddots & \vdots \\ k_{(\ell-1)j_2} & \cdots & k_{(\ell-1)j_\ell} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_{(\ell-1)} \end{bmatrix}.$$

Since the kernel matrix on the RHS is generated from the Gaussian kernel, from [54], it's known that every principal minor of a Gaussian kernel matrix is invertible, which implies that $\mathcal{O}_\Upsilon$ cannot be observable. ■

We will give a concrete example to build intuition regarding this lower bound below. For now, we note the following:

Proposition 3: Given the conditions stated in Proposition 2, it is possible to construct a measurement map $\widetilde{K} \in \mathbb{R}^{\ell \times M}$ for the system given by (3), such that the pair $(\widetilde{K}, \widehat{A})$ is observable.

Proof: The construction of the measurement map $\widetilde{K}$ is based on the rational canonical structure of $\widehat{A}^T$, which decomposes $\mathcal{V}$ into $\widehat{A}^T$-cyclic direct summands such that $\mathcal{V} = \mathcal{V}_1 \oplus \cdots \oplus \mathcal{V}_\ell$, where ℓ is the cyclic index of $\widehat{A}$. Let ξ_v be the minimal polynomial (m.p.) of v (relative to $\widehat{A}^T$): it is then the unique monic polynomial of least degree such that $\xi_v(\widehat{A}^T)v = 0$. Let $\alpha_1(\lambda)$ be the m.p. of $\widehat{A}^T_{|\mathcal{V}_1}$: then $\deg(\alpha_1(\lambda)) < M$. By the rational canonical structure theorem [53], there exists a vector $\widehat{v}_1$, such that $\xi_{v_1}(\lambda) = \alpha_1(\lambda)$. Similarly there exists a vector $\widehat{v}_2$, such that $\xi_{v_2}(\lambda) = \alpha_2(\lambda)$,

where $\alpha_2(\lambda)$, is the minimal polynomial of $\widehat{A}^T_{|\mathcal{V}_2}$ and so on. Thus we can obtain ℓ such vectors that form the measurement map $\widetilde{K} = [\widehat{v}_1, \widehat{v}_2, \cdots, \widehat{v}_\ell]^T$. Construction of these vectors $\widehat{v}_i$, can be simplified by first performing the Jordan decomposition as $\widehat{A}^T = P\Lambda P^{-1}$. Then the vectors $\widetilde{v}_i,\ i \in \{1, \ldots, \ell\}$ for Λ, can be constructed such that the entries corresponding to the leading entries of Jordan blocks of $\Lambda_{|\mathcal{V}_i}$ are nonzero. Such a construction ensures that the m.p. of vector $\widetilde{v}_i$ w.r.t $\Lambda_{|\mathcal{V}_i}$, is also the corresponding m.p. of $\Lambda_{|\mathcal{V}_i}$. Finally, the required map can be obtained as $\widetilde{K} = [\widetilde{v}_1, \widetilde{v}_2, \ldots, \widetilde{v}_\ell]^T P^{-1}$. ■

The construction provided in the proof of Proposition 3 is utilized in Algorithm 1, which uses the rational canonical structure of $\widehat{A}$ to generate a series of vectors $v_i \in \mathbb{R}^M$, whose iterations $\{v_1, \ldots, \widehat{A}^{m_1-1}v_1, \ldots, v_\ell, \ldots, \widehat{A}^{m_\ell-1}v_\ell\}$ generate a basis for $\mathbb{R}^M$. Unfortunately, the measurement map $\widetilde{K}$, being an abstract construction unrelated to the kernel, does not directly select $\mathcal{X}$. We will show how to use the measurement map to guide a search for $\mathcal{X}$ in Remark 2. For now, we state a sufficient condition for observability of a general system.

Theorem 1: Suppose that the conditions in Proposition 1 hold, with the relaxation that the Jordan blocks $\begin{bmatrix}\Lambda_1 & \cdots & \Lambda_O\end{bmatrix}$ may have repeated eigenvalues. Let ℓ be the cyclic index of $\widehat{A}$. Define:

$$\mathbf{K} = \begin{bmatrix} K^{(1)^T} & \ldots & K^{(\ell)^T} \end{bmatrix}^T \tag{8}$$

as the *ℓ-shaded matrix* (see Figure 4(b)) which consists of ℓ shaded matrices with the property that any subset of ℓ columns in the matrix are linearly independent from each other. Then system (3) is observable if Υ has distinct values, and $|\Upsilon| \geq M$.

Proof: A cyclic index of ℓ for this system implies that there exists an eigenvalue λ that's repeated ℓ times. We prove the theorem for repeated eigenvalues of dimension 1: the same statement can be proven for repeated eigenvalues for Jordan blocks using the ideas in the proof of Proposition 1. WLOG, let $\mathbf{K}$ have ℓ fully shaded, linearly independent rows, and, assume that the column indices corresponding to this eigenvalue are $\{1, 2, \ldots, \ell\}$. Define: $\boldsymbol{\lambda}_i := \begin{bmatrix}\lambda_i^{\tau_1} & \lambda_i^{\tau_2} & \cdots \lambda_i^{\tau_L}\end{bmatrix}^T$. Then,

$$\mathcal{O}_\Upsilon := \begin{bmatrix} k_{11}\boldsymbol{\lambda}_1 & k_{12}\boldsymbol{\lambda}_2 & \cdots & k_{1M}\boldsymbol{\lambda}_M \\ \vdots & \vdots & \ddots & \vdots \\ k_{\ell 1}\boldsymbol{\lambda}_1 & k_{\ell 2}\boldsymbol{\lambda}_2 & \cdots & k_{\ell M}\boldsymbol{\lambda}_M \end{bmatrix}.$$

Let $\boldsymbol{\lambda}_1 = \boldsymbol{\lambda}_2 = \cdots \boldsymbol{\lambda}_\ell := \boldsymbol{\lambda}$. Focusing on these first ℓ columns of this matrix, this implies that

we need to find constants $c_1, c_2, \ldots, c_{\ell-1}$ such that:

$$\begin{bmatrix} k_{11} \\ \vdots \\ k_{\ell 1} \end{bmatrix} = c_1 \begin{bmatrix} k_{12} \\ \vdots \\ k_{\ell 2} \end{bmatrix} + \cdots + c_{\ell-1} \begin{bmatrix} k_{1\ell} \\ \vdots \\ k_{\ell\ell} \end{bmatrix}.$$

However, these columns are linearly independent by assumption, and thus no such constants exist, implying that $\mathcal{O}_\Upsilon$ is observable. ■

While Theorem 1 is a quite general result, the condition that any ℓ columns of $\mathbf{K}$ be linearly independent is a very stringent condition. One scenario where this condition can be met with minimal measurements is in the case when the feature map $\widehat{\psi}(x)$ is generated by a dictionary of atoms with the Gaussian RBF kernel evaluated at sampling locations $\{x_1, \ldots, x_N\}$ according to (2), where $x_i \in \Omega \subset \mathbb{R}^d$, and x_i are sampled from a non-degenerate probability distribution on Ω such as the uniform distribution. For a semi-deterministic approach, when the dynamics matrix $\widehat{A}$ is block-diagonal, we can utilize a simple heuristic:

Remark 2: Let Ω be compact, $\mathcal{C} = \{c_1, \ldots, c_M\}$, $c_i \in \Omega$, and let the approximate feature map be defined by (2). Consider the system (3) with $\widehat{A} = \Lambda$, and let $\Upsilon = \{0, 1, \ldots, M-1\}$. Then the measurement map $\widetilde{K}$'s values lie in $\{0, 1\}$; in particular, each row $\widetilde{K}^{(j)}$, $j \in \{1, \ldots, \ell\}$, corresponds to a subspace $\widehat{\mathcal{H}}_j$, generated by a subset of centers $\mathcal{C}^{(j)} \subset \mathcal{C}$. Generate samples $x_i^{(j)}$ to create a kernel matrix $K^{(j)}$ that is shaded only with respect to centers $\mathcal{C}^{(j)}$. Once this is done, move on to the next subspace $\widehat{\mathcal{H}}_{j+1}$. When all ℓ rows of $\widetilde{K}$ are accounted for, construct the matrix $\mathbf{K}$ as in (8). Then the resulting system $(\mathbf{K}, \widehat{A})$ is observable.

This heuristic is formalized in Algorithm 2. Note that in practice, the matrix $\widehat{A}$ needs to be inferred from measurements of the process f_τ. If no assumptions are placed on $\widehat{A}$, it's clear that at least M sensors are required for the system identification phase. Future work will study the precise conditions under which system identification is possible with less than M sensors.

Discussion of Theoretical Results

The systems-theoretic approach taken in this paper reveals something rather surprising: functions with complex dynamics (with a small cyclic index) can be recovered with less sensor placements than functions with simpler dynamics. Although seemingly counterintuitive, it becomes clear that this is because complex dynamics, which are characterized by a lower geometric multiplicity of the eigenvalues, ensure that the orbit $\mathbf{\Theta} := \{\widehat{A} w_\tau\}_{\tau \in \Upsilon}$ traverses a greater portion of $\mathbb{R}^M \equiv \widehat{\mathcal{H}}$ and thus that fewer sensors can recover more geometric information. On the other hand, in 'simpler' functional evolution, $\mathbf{\Theta}$ evolves along strict subspaces of $\mathbb{R}^M$, and so more independent sensors are required to infer the same amount of information.

In the case described in Remark 2, we have a set of centers $\mathcal{C} = \{c_1, \ldots, c_M\}$, which generate the bases $\mathcal{F}^{\mathcal{C}} = \{\psi(c_1), \cdots, \psi(c_M)\}$. Let the cyclic index be ℓ: this implies that there exist ℓ subsets $\Psi^{(i)}$ of $\mathcal{F}^{\mathcal{C}}$ with at least one element $\psi(c_j)$ each, leading to $\binom{M}{\ell}$ possible choices: Figure 8 represents these choices as hyperplanes separating the subsets. The measurement map described in Alg. 1 in [21] induces this *decomposition of bases* $\mathcal{F}^{\mathcal{C}} = \{\Psi^{(1)}, \ldots, \Psi^{(\ell)}\}$ in polynomial time. Further, each subset $\Psi^{(i)}$ is directly associated to a subset of centers $\mathcal{C}^{(i)} \subset \mathcal{C}$, which allows us to pick targeted sensor locations $x_i \in \Omega$. In particular, for radially symmetric kernels such as the Gaussian, the centroid of the convex hull of $\mathcal{C}^{(i)}$ is sufficient for generating a sensor placement. The measurement map is a significant theoretical insight into sensor placement for dynamically changing environments, because it directly takes into account the dynamics of the process. Of course, in practice, this may be too expensive for approximate feature spaces with M very large, so one can use random sampling to generate the sensor locations instead, at the cost of N being larger than ℓ. The advantage here though is that since random sampling is computationally inexpensive, different choices of sensor placements can be generated and evaluated relatively quickly.

Another point to note is that since the collection of bases $\{\widehat{\psi}_i(x)\}_{i=1}^{M}$ determines the richness of the function space $\widehat{\mathcal{H}} \approx \mathcal{H}$ we operate in, it determines the fidelity of the model approximation to the true time-varying function. As a consequence, observability of the system in $\widehat{\mathcal{H}}$ refers to the best possible approximation in $\widehat{\mathcal{H}}$. The greater the number of bases, the higher the dimensionality, which results in greater model fidelity, but which may require a much greater number of measurements for state recovery. This is where the lower bounds presented in the paper are particularly useful, because they show that for functional evolutions corresponding to certain $\widehat{A}$, *the number of sensor placements are essentially independent of the dimensionality* M, but depend rather on the cyclic index of $\widehat{A}$.

Figure 7 gives an overall picture on the process of generating a kernel observer, while Figure 8 gives two approaches to sensor selection in our framework. The measurement map approach can generate a smaller set of sensors than the random placement approach, but comes at an additional computational cost.

Random Sensor Placement

We now elaborate on how the challenging problem of sensor placement can be tackled through random selection. This process of random selection is a product of the kernel observer model described above. We present the theoretical background required to prove Theorem 2, which states the expected number of randomly placed sensors required to monitor a given spatiotemporal process, and Theorem 3, which determines the probability with which optimal

sensor placement is ensured, given that N number of sensors have been placed.

As discussed earlier, we work with an approximate feature space $\widehat{\mathcal{H}}$, with the corresponding transition operator $\widehat{A} : \widehat{\mathcal{H}} \to \widehat{\mathcal{H}}$ representing finite-dimensional functional evolution. To achieve observability for the pair $(\widehat{A}, K)$, row vectors of the corresponding observability matrix $\mathcal{O}$, should form the basis for the $\mathbb{R}^M$-dimensional space $\widehat{\mathcal{H}}$. According to the rational canonical structure Theorem [53], $\widehat{A}$ can successively decompose the dual space $\mathbb{R}^M$ into subspaces, $\mathcal{V}_i \subset \mathcal{V}$, $i \in \{1, \ldots, \ell\}$, which have the properties: i) $\mathcal{V} = \mathcal{V}_1 \oplus ... \oplus \mathcal{V}_\ell$, ii) $\widehat{A}\mathcal{V}_i \subset \mathcal{V}_i$, and iii) $\widehat{A}|_{\mathcal{V}_i}, i \in \{1, \ldots, \ell\}$, are cyclic. The integer ℓ is unique and is called the *cyclic index of* $\widehat{A}$. Each of these properties contribute towards the theorem on the number of random samples required to achieve observability. The first property shows that the space $\mathbb{R}^M$ can be decomposed into ℓ independent subspaces. The second property shows that the vector $v_i \in \mathcal{V}_i$ stays in $\mathcal{V}_i$ even when operated upon by $\widehat{A}$. To generate bases for $\mathbb{R}^M$, one needs at least ℓ vectors $v_1, \ldots, v_\ell$, where each v_i arises from a correponsing subspace $\mathcal{V}_i$, from the set of subspaces $\mathcal{V}_1, \ldots, \mathcal{V}_\ell$. This holds due to the third property, but requires that the vectors $v_1, \ldots, v_\ell$, are the cyclic generators of their corresponding subspaces. Our analysis is based on whether a randomly selected sensor can generate a cyclic generator. To examine this, recall that a row vector $K_{(i)}$ generated by a randomly selected sensor location x_i takes the form

$$K_{(i)} = \Big[k(x_i, c_1), \ldots, k(x_i, c_M)\Big]. \tag{9}$$

Here, for radial kernels for example, the entries corresponding to the centers closer to x_i tend to be non-zero, whereas the others tend to be zero.

Overall, our construction is as follows: for each subspace $\mathcal{V}_i$, let $\mathcal{C}_{\mathcal{V}_i} \subset \mathcal{C}$ be the centers corresponding to those leading entry of Jordan blocks: then the minimum number of random samples required to generate the bases for $\mathcal{V}_i$ is equal to the number of Jordan blocks comprising $\mathcal{V}_i$. Altogether, the minimum number of random samples required to generate a basis for $\mathbb{R}^M$ is equal to the total number of Jordan blocks in $\widehat{A}$. Let ς be the total number of Jordan blocks in $\widehat{A}$. Then

$$\varsigma = \sum_{\lambda \in \sigma(\widehat{A})} \gamma_{\widehat{A}}(\lambda), \tag{10}$$

where $\sigma(\widehat{A})$ represents the spectrum of $\widehat{A}$, whose elements are the eigenvalues of $\widehat{A}$, and $\gamma_{\widehat{A}}(\lambda)$ is the geometric multiplicity corresponding to the eigenvalue λ, which is also equal to the total number of Jordan blocks corresponding to the eigenvalue λ. Define a set of centers $\mathcal{C}_\varsigma$ with elements $\{c_1, c_2, \ldots, c_\varsigma\}$, to be the centers corresponding to the leading entries of the Jordan blocks. For sensor location $x \in \Omega$, and $\epsilon > 0$, let $k(x, c_j) > \epsilon$. Denote the region $\Omega_j \subset \Omega$, such that the kernel evaluation with respect to center c_j is greater than ϵ, that is

$\Omega_j \equiv \{x \in \Omega : k(x, c_j) > \epsilon\}$. We define p_ϵ as:

$$p_\epsilon = \min_{c_j \in \mathcal{C}_\varsigma} \frac{\nu(k(x, c_j) > \epsilon)}{\nu(\Omega)}, \tag{11}$$

where ν is a measure in the real analysis sense. Hence, p_ϵ corresponds to a lower bound on the probability that a random sample lies within the $\epsilon-$shaded region of a particular center c_j. With all of this in place, we can prove the following theorem.

Theorem 2: Given the spatiotemporal function $f(x, \tau)$ with $x \in \Omega \subseteq \mathbb{R}^D, \tau \in \mathbb{Z}^+$ its kernel observer model (3), and a tolerance parameter $\epsilon > 0$, the expected number of randomly placed sensor locations required to achieve observability for the pair $(K, \widehat{A})$ is ς/p_ϵ where ς is the summation over geometric multiplicities of each $\lambda \in \sigma(\widehat{A})$ given by (10).

Proof: For each random sample, the probability that it lies within the $\epsilon-$shaded region of a particular center $c_j \in \mathcal{C}_\varsigma$ is at least p_ϵ. The series of random samples can be considered as Bernoulli trials in which p_ϵ is the probability of a successful outcome. Note this is assuming the worst case scenario that the intersection between any two ϵ-shaded region of centers belonging to the set $\mathcal{C}_\varsigma$ is empty. Observability for the pair $(K, \widehat{A})$ is achieved after ς successful outcomes are obtained, because each success ensures a row vector with non-zero entry corresponding to the leading entry of the Jordan block.

Let $X_1, X_2, \ldots, X_N$ be i.i.d. random variables whose common distribution is the Bernoulli distribution with parameter p_ϵ. The random variable $X = X_1 + X_2 + \cdots + X_N$ denotes the number of success after a certain number N of random samples. Since each X_i has the Bernoulli distribution, X will have a binomial distribution,

$$P(X = h) = \binom{N}{h} p_\epsilon^h (1 - p_\epsilon)^{N-h},$$

in which h is the number of success. The expectation of the binomial distribution, that is, the expected number of success, is Np_ϵ, and thus the expected number of trials required will be $N = \varsigma/p_\epsilon$. ■

Theorem 3: Given the spatiotemporal function $f(x, \tau)$ with $x \in \Omega \subseteq \mathbb{R}^D, \tau \in \mathbb{Z}^+$, its kernel observer model (3), a tolerance parameter $\epsilon > 0$, summation over geometric multiplicities of each $\lambda \in \sigma(\widehat{A})$ denoted by ς as in (10), and a constant $\delta \in (0, 1]$, the probability that pair $(K, \widehat{A})$ is unobservable after the selection of N random sensors is at most $e^{\frac{-1}{2}(Np_\epsilon - 2\varsigma)}$, where p_ϵ is given by (11) and $N > 2\varsigma/p_\epsilon$.

Proof: The random variable X from the proof of Theorem 1 has a binomial distribution, which enables the application of a Chernoff-type bound on its tail probabilities. A well known result on multiplicative Chernoff bound [55] is directly applied to establish this Theorem. If X

is binomially distributed, $\delta \in (0, 1]$, and $\mu = \mathbb{E}[X]$, then $P[X \leq (1-\delta)\mu] \leq \exp(-\mu\delta^2/2)$, where $\delta := 1 - \frac{\varsigma}{Np_\epsilon}$. The expression in the exponent can be simplified to $-\frac{1}{2}Np_\epsilon + \varsigma - \frac{\varsigma^2}{2Np_\epsilon}$, using $\mu = Np_\epsilon$. Note that $e^{\frac{-\varsigma^2}{2Np_\epsilon}} \leq 1$. This implies that,

$$\exp(-\mu\delta^2/2) = e^{-\frac{1}{2}(Np_\epsilon - 2\varsigma)}.e^{\frac{-\varsigma^2}{2Np_\epsilon}} \leq e^{-\frac{1}{2}(Np_\epsilon - 2\varsigma)}$$

Note that $(1-\delta)\mu = \varsigma$, and so we obtain that $P[X \leq \varsigma] \leq e^{-\frac{1}{2}(Np_\epsilon - 2\varsigma)}$. ■

For the case when $\widehat{A} \neq \Lambda$, a change of basis can be used to obtain $\Lambda = P^{-1}\widehat{A}P$, where P is the projection map. There are two challenges in performing the above analysis for Λ so obtained: first, the leading entries of Jordan blocks do not directly correspond to the centers $\{c_1, \ldots, c_M\}$ which was the case for $\widehat{A} = \Lambda$. Second, although we can obtain the transformation of the row vector (9) using the projection map P, we can no longer arrive at the definition of the probability p_ϵ as in (11). The existence of the similarity transform hints that the results in Theorems 2-3 should hold for any $\widehat{A}$, but the mathematical tools utilized in the paper seem to be insufficient to prove them. However, we present some empirical evidence for these claims for when $\widehat{A} \neq \Lambda$ in the empirical results section.

Generalizing Across Similar Spatiotemporally Evolving Systems

Building on the Kernel Observers method, let us introduce Evolving Gaussian Processes (E-GP). The primary novelty in this method of generating a model is learning an $\widehat{A}$ matrix for *multiple* systems. The ultimate goal of this research would be to generate highly efficient machine learning models that can be used instead of the costly numerical simulations for design and autonomy purposes. This would be a major success for the design and control of complex physical systems, such as soft robotics, as they would significantly reduce the cost and resources required in simulations. The ability to generalize across different physical situations, is critical. This is a difficult problem, as it requires that the model have the capability to actually learn the underlying physics and not just input-output relationships. For example, in the context of fluid flows, these models must be able to predict fluid dynamics at different conditions (such as Reynolds number) than the training data. E-GP, as far as the authors know, was the first machine learning method to generalize across spatiotemporally evolving systems of such complexity using end-to-end data.

We found that the class of functional evolutions $\mathbb{F}$ defined by linear Markovian transitions in a RKHS is still sufficient to model the nonlinear Navier Stokes equations which govern fluid dynamics, since the unknown map ψ allows us to model highly nonlinear dynamics in the input space. However, we do expect that phenomena such as bifurcation or turbulence will require nonlinear mappings $\mathcal{H}$. There are three steps to generate an E-GP model:

1) After picking the kernel and estimating the bandwidth hyperparameter σ (we utilize the maximum likelihood approach, although other approaches can be used), find an optimal basis vector set $\mathcal{C}$ using the algorithm in [56].
2) Use Gaussian process inference to find weight vectors for each time-step in the training set(s), generating the sequence $w_\tau, \tau = 1, \ldots, T$ for each system. A uniform time-step makes the next step easier but can be worked around for non-uniform data sets
3) Using the weight trajectory, use matrix least-squares with the equation $\widehat{A}[w_1, w_2, ..., w_{T-1}] = [w_2, w_3, ..., w_T]$ to solve for $\widehat{A}$.
4) To generate a multi-system model, concatenate the weight trajectories from each similar system in the least-squares computation of $\widehat{A}$. That is, let $W_\theta = [w_1^{(\theta)}, w_2^{(\theta)}, ..., w_{n-1}^{(\theta)}]$ and $W'_\theta = [w_2^{(\theta)}, w_3^{(\theta)}, ..., w_n^{(\theta)}]$ be the weight trajectory and next weight trajectory for some parameter . Then we solve the least-squares problem $\widehat{A}[W_{\theta_1}, \ldots, W_{\theta_n}] = [W'_\theta, \ldots, W'_{\theta_n}]$

For the sake of defining when it is appropriate to expect this method to be able to generalize across different spatiotemporally evolving systems, we shall define what it means for two fluid flows to be *similar*. In configuring a fluid dynamics simulation, a set of quantifiable parameters are defined. Two dynamical fluid systems S_1 and S_2 are considered *similar* if they have the same configuration of parameters and differ only in the value of at most one parameter. Furthermore, we require that the parameter be continuously variable, and that any observable quantity in the domain of the system vary smoothly as that parameter varies from its value in S_1 to its value in S_2. For example, for fluids flowing past identical cylinders, the Reynolds number associated with the free stream velocity may be varied to produce similar systems. However, to replace the system's cylinder with a triangle would be to qualitatively change the configuration of the system parameters, and thus would produce a non-similar system.

Unlike neural networks, the weights in an E-GP do not exist in some abstract, difficult-to-comprehend space, but are associated with kernel centers in specific locations in the domain. We refer to this attribute of E-GPs as the *spatial encoding* property. This property is an extremely valuable tool for gaining insight into how the learned model works. Some examples:

1) By plotting which kernel centers are associated with which invariant subspaces in the transition matrix, one can visualize where the eigenfunctions are found and how the dynamic modes are separated spatially.
2) By plotting arrows from center c_j to c_i for each of the largest elements $\hat{a}_{ij}$ of $\widehat{A}$, one can visualize how different areas of the domain influence each other's evolution.
3) By performing an eigendecomposition of the $\widehat{A}$ matrix, and transforming the eigenvectors back from the weight space to the function space, one can obtain the Koopman modes (and associated eigenvalues) of the system (see next section).

Spectral Analysis of Evolving Gaussian Process Model

We would be remiss not to examine our methods in light of Koopman operator theory, which has served as a major source of inspiration for number of new methods in analyzing dynamical systems in the last decade, especially within the Computational Flow Dynamics (CFD) community. Here we present results which show a direct connection between the Koopman modes, eigenfunctions, and eigenvalues and the spectral decomposition of the transition matrix in our model. In fact, in our final theorem, we show that the eigenvalues of our model are a subset of the eigenvalues of the infinite-dimensional Koopman operator, the eigenvectors transformed to the input space are congruent in shape to the Koopman modes, and the eigenfunctions are identical. These results have allowed us to construct new methods for sensor placement under even more difficult conditions than described previously – specifically, the situation of one (or a few) sensors attached to moving agents or robots.

For a general dynamical system $f_{\tau+1} = \mathbb{F}(f_\tau)$ defined on a state space (for us, an RKHS $f \in \mathcal{H}$), we can define an arbitrary, vector-valued *observable* $g : \mathcal{H} \to \mathbb{R}^N$. Note that the space of observables g is a vector space. The Koopman operator U is defined to be the operator on the space of observables such that:

$$Ug(f_\tau) = g(\mathbb{F}(f_\tau)) = g(f_{\tau+1}) \tag{12}$$

This operator is clearly linear from its definition, and thus it is reasonable to examine its spectral properties. The special observables $\phi : \mathcal{H} \to \mathbb{C}$ that have the property

$$U\phi(f_\tau) = \phi(\mathbb{F}(f_\tau)) = \phi(f_{\tau+1}) = \lambda\phi(f_\tau) \tag{13}$$

are the eigenfunctions of U, and the associated λ are the eigenvalues. Suppose now we have a vector-valued observable $u(f, x)$, where $x \in \Omega$ and $f \in \mathcal{H}$.

Definition 2: The Koopman mode $s(x)$ at isolated eigenvalue λ of algebraic multiplicity 1 is the projection of $u(f, x)$ onto the eigenfunction $\phi_\lambda(f)$ of U at λ. [57]

As previously shown by Rowley in 2009 [58], the modes produced by the Dynamic Mode Decomposition (DMD) algorithm constitute a subset of Koopman modes. Mezic (2013) [57] showed that there exists, in principle if not in practice, a rigorous method for computing the full set of Koopman modes by a method known as generalized Laplace analysis (GLA). We have generated a number of results interpreting the E-GP model in terms of Koopman operator theory, culminating in the proof that the eigenvalues and eigenvectors of $\widehat{A}$ are related to the Koopman eigenvalues and modes.

Proposition 4: Let $\mathcal{H}$ be a RKHS with an approximate feature space $\widehat{\mathcal{H}}$, and let $u(f, x)$

be an observable in the Koopman sense with respect to the dynamical system $f_{\tau+1} = \mathbb{F}(f_\tau)$ in $\mathcal{H}$. Then $\hat{u}(f,x) \coloneqq u(\widehat{f},x)$, where $\widehat{f}$ is the projection of $f \in \mathcal{H}$ onto $\widehat{\mathcal{H}}$, is also an observable.

This follows from the fact that the projection from the function space to its subspace is well-defined when the $\widehat{\psi}_i$ are independent. Now, Koopman modes of observables are of interest because they are akin to the eigenvector expansions utilized in linear dynamics. If we separate the Koopman operator into $U = U_s + U_r$ where U_s has a pure point spectrum with k points and U_r has a pure continuous spectrum, then we can write,

$$U^t u(f,x) = u^*(f,x) + \sum_{j=1}^{k} \lambda_j^t \phi_j(f) s_j(x) + U_r^t u(f,x) \tag{14}$$

where $u^*(x)$ represents the time-averaged of part of the field, which corresponds with $\lambda = 1$ (see GLA in [57]). The continuous-spectrum component is usually discarded/neglected since it represents the part of the field that is genuinely aperiodic (or chaotic) in time, which could be modeled as a stochastic process [57]. Now, from this expansion we can prove that:

Proposition 5: Let $u(f,x)$ and $\hat{u}(f,x)$ be observables as in Proposition 4, and $s_j(x)$ be the Koopman modes associated with the projection of the former onto the eigenfunctions ϕ_j of U at λ_j. Then the Koopman modes associated with $\hat{u}$ are:

$$\hat{s}_j(x) = \frac{\phi_j(\widehat{f})}{\phi_j(f)} s_j(x) \tag{15}$$

Proof: Using the spectral expansion and the definition of the observables, we have:

$$U^t \hat{u}(f,x) = U^t u(\widehat{f},x)$$

$$\hat{u}^*(f,x) + \sum_{j=1}^{k} \lambda_j^t \phi_j(f) \hat{s}_j(x) + U_r^t \hat{u}(f,x) = u^*(\widehat{f},x) + \sum_{j=1}^{k} \lambda_j^t \phi_j(\widehat{f}) s_j(x) + U_r^t u(\widehat{f},x)$$

Since this must be true for all t, the terms must match, and the hypothesis follows. ■

Note that this means the modes of the two observables are identical in shape (only differing by a multiplicative constant). In the exceptional case that $\phi_j(f) = 0$, the GLA method is unable to compute $s_j(x)$ anyway, so that is not an issue for this proposition. The final step in our analysis is to connect the Koopman modes with the spectral decomposition of our model's $\widehat{A}$ operator in the dual space.

Theorem 4: In an E-GP or KO model of the dynamical system $f_{\tau+1} = \mathcal{A} f_\tau$,

1) The eigenvalues of $\widehat{A}$ are a subset of the eigenvalues of the Koopman operator.
2) The eigenfunctions of the Koopman operator in the dual space correspond with a subset of the eigenfunctions of the Koopman operator on f_0, that is $\hat{\phi}_j(w_\tau) = \phi_j(f_\tau)$.

3) If the observable $u(f,x)$ is the evaluation operator, $u(f,x) \coloneqq f(x)$, then the eigenvectors v_j of $\widehat{A}$ are related to the Koopman modes of the observable by:

$$s_j(x) = \frac{\phi_j(f)}{\phi_j(\widehat{f})}\widehat{\Psi}(x) \cdot v_j \tag{16}$$

where $\widehat{\Psi}(x)$ is the feature map of our model as a row vector.

Proof: It should be clear that, as long as the projection onto the approximate feature space is well defined, that $P(f_\tau) \coloneqq w_\tau$ is an observable. Then if we let $\phi_j(f) = \langle P(f), q_j \rangle$, where q_j are eigenvectors of the adjoint $\widehat{A}^*$ (that is, $\widehat{A}^* q_j = \bar{\lambda}_j q_j$), then ϕ_j is an eigenfunction of the Koopman operator in $\mathcal{H}$ since $U\phi_j(f_\tau) = \langle P(f_{\tau+1}), q_j \rangle = \langle w_{\tau+1}, q_j \rangle = \langle \widehat{A} w_\tau, q_j \rangle = \langle w_\tau, \widehat{A}^* q_j \rangle = \lambda_j \langle w_\tau, q_j \rangle = \lambda_j \phi_j(f_\tau)$. This shows that:

1) The eigenvalues of $\widehat{A}$ correspond with that of the Koopman operator.
2) The eigenfunctions of the Koopman operator in the dual space, known to be $\hat{\phi}_j(\cdot) = \langle \cdot, q_j \rangle$, correspond with the eigenfunctions of the Koopman operator in $\mathcal{H}$.

In the case that $u(f,x) \coloneqq f(x)$, we have the relationship $\hat{u}(f,x) = \widehat{f}(x) = \widehat{\Psi}(x) \cdot w(f)$ according to the formulation of our model. Now, it is well known that the spectral expansion of a finite linear system is:

$$w_t = \sum_{j=1}^{M} \lambda_j^t \hat{\phi}_j(w_0) v_j$$

where v_j are the eigenvectors of the transition matrix. Therefore, we have:

$$U^t \hat{u}(f_0, x) = U^t \left(\widehat{\Psi}(x) \cdot w(f_0) \right) = \widehat{\Psi}(x) \cdot w_t = \widehat{\Psi}(x) \cdot \left(\sum_{j=1}^{M} \lambda_j \hat{\phi}_j(w_0) v_j \right) = \sum_{j=1}^{M} \lambda_j \phi_j(f_0) \widehat{\Psi}(x) \cdot v_j$$

Comparing this to the spectral expansion of $\hat{u}(f,x)$, we see (letting $\lambda_1 = 1$) that:

3) $\hat{u}^*(f,x) = \widehat{\Psi}(x) \cdot v_1$, $\hat{s}_j(x) = \widehat{\Psi}(x) \cdot v_j$, and that there is no continuous spectrum component. The hypothesis now follows from Proposition 5.

■

In the end, we have established a direct connection between the spectral decomposition of the transition matrix in the dual space of the approximate feature space and the Koopman modes, eigenfunctions, and eigenvalues.

Corollary 1: Suppose, given an $\epsilon > 0$, we are able to find an approximate feature space $\widehat{\mathcal{H}}$ such that for all $f \in \mathcal{H}$ there exists $\widehat{f} \in \widehat{\mathcal{H}}$ such that $\|f - \widehat{f}\| < \epsilon$. Then, if $\phi_j(f) \neq 0$, there

exists an approximate feature space such that $\widehat{\Psi}(x) \cdot v_j$ is arbitrarily close to the Koopman mode $s_j(x)$ (where v_j is an eigenvector of the dual space transition matrix $\widehat{A}$).

Proof: Since ϕ_j is continuous, we can make $|\phi_j(f) - \phi_j(\widehat{f})|$ arbitrarily small, which means if $\phi_j f \neq 0$ we can make $\frac{\phi_j(f)}{\phi_j(\widehat{f})}$ arbitrarily close to 1. From Theorem 4 and the fact that $s_j \in \mathcal{H}$ means s_j is bounded, we can conclude that $\widehat{\Psi}(x) \cdot v_j$ can be made arbitrarily close to $s_j(x)$ everywhere in Ω.

■

In essence, this says that if we can find a close-enough approximate feature space, then the eigenvectors of the transition matrix in the dual space correspond exactly with the Koopman modes in the input space.

Invariant Subspaces

One way of viewing the invariant subspaces concept is to say that information contained in an invariant subspace never leaves that invariant subspace. We hypothesize that the kernel centers associated with the invariant subspaces of a spatiotemporally evolving system are generally associated with spatial regions in the domain (and not just homogeneously spread all over and or mixed with the other invariant subspaces). This hypothesis makes sense both physically and mathematically. In physics, the *principle of locality* states that an object is only directly influenced by its immediate surroundings [59]. If this is the case (and there is no reason to believe that it is not, apart from certain quantum dynamics situations), and the E-GP model accurately captures the physics of the system, then information (measurable phenomena) may only travel continuously from one point in the domain to another. Mathematically, since a value at any one point in the domain is influenced by the weights of multiple nearby centers, we would expect nearby centers to be connected dynamically, unless separated by "plains" where the values are indistinguishable from noise.

In an ideal case, the Jordan form of a $n \times n$ matrix $\widehat{A}$ is block diagonal, and therefore gives a decomposition of the n dimensional Euclidean space into clearly separated invariant subspaces of $\widehat{A}$. The cyclic index, which can be found by counting the geometric multiplicities of eigenvalues in $\widehat{A}$, gives the number of invariant subspaces. In practice, data-driven approximations of $\widehat{A}$ rarely give such easily interpretable properties. This fact drives the need for an algorithm that can divide the system into invariant subspaces even when the boundaries between such spaces are not mathematically precise. This algorithm can be thought of as a 'soft' Jordan decomposition.

Each block in the Jordan normal form has a set of corresponding eigenvectors and an

eigenvalue with geometric multiplicity. When we transform the former back into the domain space, we obtain complex-valued functions which are the Koopman modes of the system. These provide an image of what kind of structures we see in the dynamics. The eigenvalues describe the frequency with which these structures oscillate between their real and imaginary forms, as well as their exponential growth or decay in magnitude.

k-Invariant Subspaces Clustering

In response to the need for an algorithm that can separate invariant subspaces in the linear model of a system generated by data, we have developed what we call "k-invariant subspaces clustering". The intuition behind our algorithm is to replace the Euclidean distance in the well-known k-means algorithm with a different metric of "nearness", namely one corresponding with the dynamic connections in the space. The $\widehat{A}$ matrix provides easy access to these: its rows $\widehat{A}_{i*}$ indicate which centers inform the i$^{\text{th}}$ value of $w_{\tau+1}$, and its columns $\widehat{A}_{*j}$ indicate what centers will be informed by the j$^{\text{th}}$ value of w_τ. However, these values do tend to be biased towards the eigenmodes with higher frequencies (that is, those whose eigenvalues have a greater polar angle) and those which grow exponentially. To control for this, we chose to modify the eigenvalues of the matrix in the following way: we (1) zeroed any eigenvalue that is clearly inside (not on) the unit circle, (2) unitized the remaining eigenvalues, and (3) adjusted the frequency of the remaining eigenvalues on the unit circle to either $\pm\frac{\pi}{4}$. If the eigen-decomposition of the original was $\widehat{A} = UDU^{-1}$, then we can reconstruct a new matrix with our modified eigenvalues $\bar{D}$ as $\bar{A} = U\bar{D}U^{-1}$.

Much like the pairwise-squared deviations of points formulation of the k-means clustering problem, we can now write our problem as:

$$\arg\max_{S} \sum_{i=1}^{k} \frac{1}{2|S_i|} \sum_{\substack{x_i \neq x_j \\ x_j, x_i \in S_i}} \bar{A}_{ij}^2$$

This problem can be solved with Algorithm 5.

Note the differences between this and the k-means clustering algorithm: first, we are *maximizing* since the terms represent influence rather than distance. Secondly, in the possible case that $|S_k| = 0$, we make the total value zero in order to avoid division by zero. Thirdly, we exclude centers' influence on themselves from the score by taking $S_k \setminus \{i\}$.

This algorithm can be repeated as many times as desired with different random initial conditions, and the result which produces the highest score retained.

Empirical Results

We begin with numerical results obtained for determining sensing locations by application of kernel observers to the prediction of spatiotemporal functions. The highlight here is a set of results on sensing global ocean temperatures with information from just a few locations using the AVHRR satellite data. This is followed by an application of E-GP in predicting spatiotemporal functions. We include results in predicting weed growth in simulated agricultural fields. The application of E-GP to model flow past a cylinder at different Reynolds numbers is explored in "Learning Fluid Flows with Evolving Gaussian Processes". We conclude the results section with a comparison of eigenvalues and Koopman modes computed from E-GP to those of DMD.

Sensing Locations for Synthetic Data Sets

The goal of this experiment is to investigate the dependency of the observability of system (3) on the shaded observation matrix and the lower bound presented in Proposition 2. The domain is fixed on the interval $\Omega = [0, 2\pi]$. First, we pick sets of points $\mathcal{C}^{(\iota)} = \{c_1, \ldots, c_{M_\iota}\},\ c_j \in \Omega$, $M = 50$, and construct a dynamics matrix $A = \Lambda \in \mathbb{R}^{M\times M}$, with cyclic index 5. We pick the RBF kernel $k(x, y) = e^{-\|x-y\|^2/2\sigma^2}$, $\sigma = 0.02$. Generating samples $\mathcal{X} = \{x_1, \ldots, x_N\}$, $x_i \in \Omega$ randomly, we compute the ℓ-shaded property and observability for this system. Figure 9(a) shows how shadedness is a necessary condition for observability, validating Proposition 1: the slight gap between shadedness and observability here can be explained due to numerical issues in computing the rank of $\mathcal{O}_\Upsilon$.

Next, we consider a system with a cyclic index $\ell = 18$ to verify random sensor placement results. We constructed the measurement operator K using the heuristic in Remark 2 (Algorithm 2), and random sensor selection to generate the sampling locations $\mathcal{X}$. These results are presented in Figure 9(b). The plot for random sampling which has been averaged over 100 runs, resembles a c.d.f function of an exponential distribution $F(X = x) = 1-\exp(-\lambda x)$. This verifies the claim made in Theorem 3, as the probability of becoming unobservable decays exponentially with the number of sensors placed. Also, fitting an exponential distribution, we found that the mean λ^{-1} comes close to the ratio ς/p, which is the expected number of randomly placed sensors required for observability as per Theorem 2. Note that observability is not achieved if the number of samples $N < \ell$, which is experimental evidence of the result in Proposition 2.

Comparison With Nonstationary Kernel Methods on Real-World Data

We use three real-world datasets to evaluate and compare the kernel observer with the two different lines of approach for non-stationary kernels discussed in section on related work. For

the Process Convolution with Local Smoothing Kernel (PCLSK) and Latent Extension of Input Space (LEIS) approaches, we compare with NOSTILL-GP [8] and [37] respectively, on the Intel Berkeley, Irish Wind and Ozone data-sets.

Model inference for the kernel observer involved three steps: 1) picking the Gaussian RBF kernel $k(x, y) = e^{-\|x-y\|^2/2\sigma^2}$, a search for the ideal σ is performed for a sparse Gaussian Process model (with a fixed basis vector set $\mathcal{C}$ selected using the method in [56]. For the data set discussed in this section, the number of basis vectors were equal to the number of sensing locations in the training set, with the domain for input set defined over $\mathbb{R}^2$; 2) having obtained σ, Gaussian process inference is used to generate weight vectors for each time-step in the training set, resulting in the sequence $w_\tau, \tau \in \{1, \ldots, T\}$; 3) matrix least-squares is applied to this sequence to infer $\widehat{A}$ (Algorithm 3). For prediction in the autonomous setup, $\widehat{A}$ is used to propagate the state w_τ forward to make predictions with no feedback, and in the observer setup, a Kalman filter (Algorithm 4) with N determined using Proposition 2, and locations picked randomly, is used to propagate w_τ forward to make predictions. We also compare with a baseline GP (denoted by 'original GP'), which is the sparse GP model trained using all of the available data.

Our first dataset, the Intel Berkeley research lab temperature data, consists of 50 wireless temperature sensors in indoor laboratory region spanning 40.5 meters in length and 31 meters in width (http://db.csail.mit.edu/labdata/labdata.html). Training data consists of temperature data on March 6th 2004 at intervals of 20 minutes (beginning 00:20 hrs) which totals to 72 timesteps. Testing is performed over another 72 timesteps beginning 12:20 hrs of the same day. Out of 50 locations, we uniformly selected 25 locations each for training and testing purposes. Results of the prediction error are shown in box-plot form in Figure 10(a) and as a time-series in Figure 10(b), note that 'Auto' refers to autonomous set up. Here, the cyclic index of $\widehat{A}$ was determined to be 2, so N was set to 2 for the kernel observer with feedback. Note that here, even the autonomous kernel observer outperforms PCLSK and LEIS overall, and the kernel observer with feedback with $N = 2$ significantly outperforms all other methods, which is why we did not include results with $N > 2$.

The second dataset is the Irish wind dataset, consisting of daily average wind speed data collected from year 1961 to 1978 at 12 meteorological stations in the Republic of Ireland (http://lib.stat.cmu.edu/datasets/wind.desc). The prediction error is in box-plot form in Figure 11(a) and as a time-series in Figure 11(b). Again, the cyclic index of $\widehat{A}$ was determined to be 2. In this case, the autonomous kernel observer's performance is comparable to PCLSK and LEIS, while the kernel observer with feedback with $N = 2$ again outperforms all other methods.

Finally, the Ozone dataset measures ozone concentration (in parts per billion) measured at 60 stations by the United States Environmental Protection Agency [60] across USA. Due to missing measurements, we only selected data from year 1997 to 2013 for training and evaluation. For each station, we averaged ozone concentration over a period of three months, resulting in four quarters per year. Out of 60 sensor locations, we uniformly selected 30 for training and the remaining locations for testing purposes. The prediction error results are presented in box-plot form in Figure 12(a) and as a time-series in Figure 12(b). Here, the cyclic index of $\widehat{A}$ was determined to be 1. In this case, the performance of autonomous kernel observer is comparable to PCLSK and LEIS, with kernel observer with feedback with $N = 1$ performing the best. Table 1 reports the total training and prediction times associated with PCLSK, LEIS, and the kernel observer. We observed that, 1) the kernel observer is an order of magnitude faster than the competing methods, and 2) even for small sets, competing methods did not scale well.

Prediction of Global Ocean Surface Temperature

We analyzed the feasibility of our approach on a very large dataset from the National Oceanographic Data Center: the 4 km AVHRR Pathfinder project, which is a satellite monitoring global ocean surface temperature (Figure 13(a) shows the raw data). This dataset is challenging, with measurements at over 37 million possible coordinates, but with only around 3-4 million measurements available per day, which means a lot of missing data. The goal was to learn the day and night temperature models on data from the year 2011, and then to monitor the system through the year 2012. Success in monitoring would demonstrate two things: 1) the modeling process can capture spatiotemporal trends that generalize across years, and 2) the observer framework allows us to infer the state using a number of measurements that are an order of magnitude fewer than available. Note that due to the size of the dataset and the high computational requirements of the nonstationary kernel methods, a comparison with them was not pursued. To build the autonomous kernel observer and general kernel observer models, we followed the same procedure outlined in Section , but with $\mathcal{C} = \{c_1, \ldots, c_M\}$, $c_j \in \mathbb{R}^2$, $|\mathcal{C}| = 300$. The Kalman filter for the general kernel observer model used $N \in \{250, 500, 1000\}$ at random locations to track the system state given a random initial condition w_0. As a fair baseline, the observers are compared to training a sparse GP model (labeled 'original') on approximately $400,000$ measurements per day. Figure 13(b) is an estimate of global ocean surface temperatures obtained using autonomous kernel observer. Figures 13(c) and 13(d) compare the autonomous and feedback approach with $1,000$ samples to the baseline GP; here, it can be seen that the autonomous does well in the beginning, but then incurs an unacceptable amount of error when the time series goes into 2012, i.e. where the model has not seen any training data, whereas FKO does well throughout. Figures 13(e) and 13(f) show a comparison of the RMS error of estimated values from the real data. This figure

shows the trend of the observer getting better and better state estimates as a function of the number of sensing locations N. Note that we checked the performance of training a GP with only $1,000$ samples as a control, but the average error was about 10 Kelvins, i.e. much worse than FKO. The time required for using the kernel observer is significantly less than retraining the model every time step; see Figures 13(g)-13(h).

Weather Anomaly in 2012: We further investigated the poor performance of the autonomous kernel observer in the year 2012 as observed in Figures 13(c) and 13(d). Clearly, the prediction error blows up at the start of May in the year 2012, indicating that the autonomous model trained using the data from 2011 does well in capturing the annual weather dynamics up to the month of May 2012. We turned our attention towards the weather in May 2012, as changes in ocean temperatures are directly related to the weather. As we guessed, severe weather conditions were reported on the east coast of United States in May 2012, and this anomaly continued over the period of May to June 2012. Here, the apparent poor performance of autonomous kernel observer can actually be a useful indicator in detecting the anomalous behavior, as was the case with the ocean temperature in May 2012 which had deviated from the normal weather dynamics observed in the year 2011 in which no severe weather anomalies were reported. Further, we identified the locations at which the prediction error was two standard deviations above the mean error. These error locations are plotted in Figure 14. Error locations on 6-7th May coincide with the severe weather onset on the east coast of United states and that of 28-29th May were along the track of storm Beryl approaching the eastern coastline. This shows that the autonomous kernel observer model captures the dynamics of spatiotemporal evolution, and has the potential of identifying current behavior as anomalous to that observed in the training set.

Predicting evolution of weed density in agricultural fields

As a proof-of-concept in applying our methods for learning dynamics and/or inferring the state of spatiotemporally-evolving systems, we examined the problem of predicting weed growth in agricultural fields.

Past work has presented detailed analysis of weed growth models, in which measurements of seed bank density for various species of weeds were conducted [61]–[64]. In order to generate data upon which to test our methods, we implemented a weed growth simulation model whose rate of seedling emergence agrees with that found in the above research [1]. A Poisson process is utilized to simulate the temporal evolution of emergence events. This assumption is reasonable over the short time scales in which robots may fully weed a field.

Other work in the field of crop science [65], [66] has shown that the spatial variation in the seed bank density for some species of weeds may be modeled via the Gini Coefficient of Concentration (GCC). This model is accurate for common waterhemp (*Amaranthus tuberculatus*), and thus we found it useful in our simulation of the spatial distribution of the seed bank density. For more reading. see [1], [67], [68] regarding the relationship between seed bank emergence patterns and environmental conditions such as temperature and moisture.

The weed growth simulation is based on Bernoulli random variables, operating on a matrix of cells, each 0.8 m^2, comprising a gridded field of 0.4 hectares, or a cellular automata model [69]. Seeds emerge from a limited seed bank, forming a binomial distribution over time. The parameters used, summarized in Table 2, are aligned with the growth model for common waterhemp determined in [65]. The initial density of the seed bank in each cell is S_0 on average (between 600 and 1560 seeds per cell). However, at the start of the simulation, the seed bank density in each cell, $S_0(x, y)$, is chosen so that Gini Coefficient of Concentration (GCC) between all the cells, used to ensure the relative density of weeds aligns with that seen in real experiments, is from 0.31 to 0.35, as was determined experimentally in [65]. The field is first divided into fifty patches of weeds, with centers chosen uniformly at random, and sizes from zero to twenty cells in each dimension. Each cell has an initial density between zero and twenty percent of S_0, chosen uniformly at random. Finally, for each patch of weeds, up to an additional S_0 weeds are added to each cell within each patch, so that the density of each patch follows a normal distribution.

Upon initialization of the simulation, a certain number of days, d_0, are allowed to elapse before weeding starts. The number of emerging weeds in each cell, N_{emerge}, is a randomly generated Poisson variable with mean, $\lambda\left(x, y, t\right)$, such that 90 percent of the seed bank, $S\left(x, y, t\right)$, emerges in T_{total}, which is two months. This emergence rate is aligned with past work [61]–[64], [68], which all present measurements of the seed bank densities for various species of weeds, and provide an analysis of weed growth models for these species.

The weed density in each cell, $\zeta\left(x, y, t\right)$, grows as seeds emerge from the seed bank. The maximum weed height at each cell, $\delta\left(x, y, t\right)$, increases from zero height at a fixed rate Γ inches per day.

The main challenge with weeding robots is that they only have access to sparse data about weed density and height, and no data about the seedbank. In fact, the robots can only sense the row that they are in and potentially the adjacent rows on either side. They have no information about the field in locations that have not been visited. Hence, the robots have to try and predict the global weed density given sparse information. This is a great application for the Kernel

observer techniques studied in this paper. To demonstrate the proof of concept, we trained a kernel observer model on the weed density data generated by our simulations, so that robots in the field can quickly infer the state of the field globally from partial measurements from a few rows.

We used Radial Basis Function (RBF) kernels with a bandwidth approximately 5 cells (4.5 meters) wide. These were centered throughout the domain using the algorithm in [70]. After training a GP on each snapshot of the data, we obtained a weight vector trajectory. We used linear least squares do determine the best linear transition in the weight space for this trajectory. Below we have included images of system as predicted by feeding forward the initial condition through this linear transition, as a comparison with the original data. The model is able to approximate the weed density growth data very well, with percent error averaging 5%. These results show the feasibility of modeling weed growth using E-GPs. Ongoing work involves a massive field campaign to collect the weed density data required to learn these models. Such datasets are currently not available, and the TerraSentia robots discussed in Sidebar: Key control problems in agriculture are being used to collect this data.

Comparison of Eigenvalues and Koopman modes with DMD

In order to empirically verify our theoretical results, we compared the eigenvalues and Koopman modes derived from $\widehat{A}$ in our E-GP model with the eigenvalues and Koopman modes derived by the well-known Dynamic Mode Decomposition (DMD) algorithm using data from fluid flowing past a cylinder at different Reynolds numbers. The results are presented in figures 16 and 17. As can be seen, the modes generated by E-GP match those generated by DMD at least as well as E-GP predictions match the original data.

Conclusion

Machine learning techniques such as Gaussian Process (GP) regression and deep learning are providing increasingly more powerful ways of learning predictive models. For all their power, however, since these techniques are limited by the datasets that they are trained with, their predictions are not always reliable when predicting real-world complex spatiotemporally-evolving phenomena with high variability; years' worth of weather data cannot be a reliable predictor of current weather, nor can data from one farm directly relate to another. Because of this, when engineering complex cyber-physical systems such as team of mechanically weeding robots, engineers have devise ways to supplement the predictions of the machine learning models with real measurements. This is not always immediate when one works in the abstract feature spaces utilized in machine learning models. In this tutorial we presented kernel observers and

their extension Evolving Gaussian Processes (E-GP), which are formed by embedding a linear dynamical system in the reproducing kernel Hilbert spaces generated by the GP model. The parameters of the linear dynamical system can be trained to approximately predict the evolution. This leads to powerful and generalizable models, as our results demonstrated in predicting solutions to the Navier-Stokes equations. When supplemented with a predict-and-correct strategy, such as a Kalman filter embedded with the linear dynamical system in the RKHS, the dynamical state of the system can be kept on track with real sensor measurements. This creates a very powerful prediction system that is capable of predicting nonlinear evolution using minimal sensor measurements. We showed that the geometric properties of the linear dynamical system can be utilized to determine sensor numbers and locations. Looking forward, it would interesting to extend the idea to include nonlinear dynamical systems in the feature space, which would improve the capability of the E-GP model. At present, our results on fairly complex datasets demonstrate that feedback with sensors creates a robust monitoring system with the predictions with a simple linear model. The theoretical limits of the tradeoff between improving the model versus increasing the sensing in the context of E-GPs is also an interesting avenue of study. Finally, also we note that deep neural networks have also been applied to the spatiotemporal modeling problem with significant empirical success [71], but because these methods are 1) nonlinear in their parameters, and 2) seem to lack the spatial-encoding properties of some types of kernels, generalizing the analysis considered here for those systems is an interesting open question. As sidebar "Feature Spaces in Machine Learning" indicates, there could be very interesting future work that generalizes the idea of embedding controllers and observers in more advanced features spaces considered in machine learning models.

Software

To facilitate implementation of the material presented in this paper we have made an open source code repository. A Matlab version is available at http://daslab.illinois.edu/software.html or at https://github.com/hkingravi/FunctionObservers with examples and documentation. In addition, a Python version is available on GitHub at https://github.com/hkingravi/funcobspy.

ACKNOWLEDGMENTS

The work presented was supported in part by AFOSR #FA9550-15-1-0146, and USDA/NSF CPS grant 2018-67007-28379, and USDA/NSF NRI grant 2019-67021-28989. We thank EarthSense Inc. (www.earthsense.co) for providing us approval to use Figures 1 and S1. The authors also thank Prof. Stephen Long, Prof. Carl Bernacchi, Prof. Michael Gore, and Prof. Edward

Buckler on insights about the phenotyping bottleneck and simulation of plants (*crops-in-silico*), Prof. Adam Davis on inputs on the herbicide-resistant weed crisis, and Prof. Sarah Lovell and Dr. Chinmay Soman on sustainable agricultural production systems and perennial polycultures. We also thank the members of the UIUC Center for Digital Agriculture for their inputs.

References

[1] Wyatt McAllistar, Denis Osipychev, Girish Chowdhary, and Adam Davis. Multi-agent planning for coordinated robotic weed killing. In *International Conference on Intelligent Robots and Systems (IROS)*, Madrid, Spain, 2018. IEEE.

[2] Tim P Barnett, David W Pierce, and Reiner Schnur. Detection of anthropogenic climate change in the world's oceans. *Science*, 292(5515):270–274, 2001.

[3] Matthew J Heaton, Matthias Katzfuss, Shahla Ramachandar, Kathryn Pedings, Eric Gilleland, Elizabeth Mannshardt-Shamseldin, and Richard L Smith. Spatio-temporal models for large-scale indicators of extreme weather. *Environmetrics*, 22(3):294–303, 2011.

[4] Noel Cressie and Christopher K Wikle. *Statistics for spatio-temporal data*. John Wiley & Sons, 2011.

[5] Carl E. Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, December 2005.

[6] K.-R. Müller, S. Mika, G. Rätsch, S. Tsuda, and B Schölkopf. An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks*, 12(2):181–202, 2001.

[7] Bernhard Schölkopf and Alexander J Smola. *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[8] Sahil Garg, Amarjeet Singh, and Fabio Ramos. Learning non-stationary space-time models for environmental monitoring. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada.*, 2012.

[9] Chunsheng Ma. Nonstationary covariance functions that model space–time interactions. *Statistics & Probability Letters*, 61(4):411–419, 2003.

[10] Christian Plagemann, Kristian Kersting, and Wolfram Burgard. Nonstationary gaussian process regression using point estimates of local smoothness. In *Machine learning and knowledge discovery in databases*, pages 204–219. Springer, 2008.

[11] Marco Todescato, Andrea Carron, Ruggero Carli, Gianluigi Pillonetto, and Luca Schenato. Efficient spatio-temporal gaussian regression via kalman filtering. *arXiv preprint arXiv:1705.01485*, 2017.

[12] G. Chowdhary, H. Kingravi, J.P. How, and P. Vela. Nonparametric adaptive control using gaussian processes. In *IEEE Conference on Decision and Control (CDC)*. IEEE, 2013.

[13] Suvrit Sra, Sebastian Nowozin, and Stephen J Wright. *Optimization for machine learning*. Mit Press, 2012.

[14] Sadeep Jayasumana, Richard Hartley, Mathieu Salzmann, Hongdong Li, and Mehrtash Harandi. Kernel Methods on Riemannian Manifolds with Gaussian RBF Kernels. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2015.

[15] A. Gelb. *Applied Optimal Estimation*. MIT press (QA402.A5), Cambridge, MA, 1974.

[16] Kanti V Mardia, Colin Goodall, Edwin J Redfern, and Francisco J Alonso. The kriged kalman filter. *Test*, 7(2):217–282, 1998.

[17] F. Cucker and S. Smale. On the mathematical foundations of learning. *Bulletin of the American Mathematical Society*, 39(1), 2002.

[18] Hassan A Kingravi, Girish Chowdhary, Patricio A Vela, and Eric N. Johnson. Reproducing kernel hilbert space approach for the online update of radial bases in neuro-adaptive control. *Neural Networks and Learning Systems, IEEE Transactions on*, 23(7):1130–1141, 2012.

[19] M. Liu, G. Chowdhary, B. Castra da Silva, S. Liu, and J. P. How. Gaussian processes for learning and control: A tutorial with examples. *IEEE Control Systems Magazine*, 38(5):53–86, Oct 2018.

[20] Carl Edward Rasmussen. Gaussian processes for machine learning. Technical report, Max Planck Institute for Biological Cybernetics, 2006.

[21] Hassan Kingravi, Harshal Maske, and Girish Chowdhary. Kernel observers: Systems theoretic modeling and inference of spatiotemporally varying processes. In *Advances in Neural Information Processing Systems (NIPS)*, Barcelona, Spain, 2016.

[22] Joshua Whitman, Harshal Maske, Girish Chowdhary, Hassan Kingravi, Chen Lu, and Balaji Jayaraman. Modeling nonlinear dynamical fluid flows with evolving gaussian process models. In *NIPS workshop on Machine Learning in the Wild*, Barcelona, Spain, 2016. NIPS.

[23] Hassan A. Kingravi, Harshal Maske, and Girish Chowdhary. Kernel controllers: A systems-theoretic approach for data-driven modeling and control of spatiotemporally evolving processes. *arXiv*, abs/1508.02086, 2015.

[24] Joshua Whitman and Girish Chowdhary. Learning dynamics across similar spatiotemporally-evolving physical systems. In *Conference on Robot Learning*, pages 472–481, 2017.

[25] Harshal Maske, Hassan Kingravi, and Girish Chowdhary. Sensor selection via observability analysis in feature space. In *American Control Conference*, 2018. accepted.

[26] Christopher K Wikle. A kernel-based spectral model for non-gaussian spatio-temporal processes. *Statistical Modelling*, 2(4):299–314, 2002.

[27] Jonathan R Stroud, Peter Muller, and Bruno Sanso. Dynamic models for spatiotemporal data. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(4):673–689, 2001.

[28] Jouni Hartikainen et al. Sequential inference for latent temporal gaussian process models. 2013.

[29] Finn Lindgren, Haavard Rue, and Johan Lindstrom. An explicit link between gaussian fields and gaussian markov random fields: the stochastic partial differential equation approach.

Journal of the Royal Statistical Society: Series B (Statistical Methodology), 73(4):423–498, 2011.

[30] Terrence T Ho, Paul W Fieguth, and Alan S Willsky. Multiresolution stochastic models for the efficient solution of large-scale space-time estimation problems. In *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*, volume 6, pages 3097–3100. IEEE, 1996.

[31] Fernando Pérez-Cruz, Steven Van Vaerenbergh, Juan José Murillo-Fuentes, Miguel Lázaro-Gredilla, and Ignacio Santamaria. Gaussian processes for nonlinear signal processing: An overview of recent advances. *Signal Processing Magazine, IEEE*, 30(4):40–50, 2013.

[32] Simo Särkkä and Robert Piché. On convergence and accuracy of state-space approximations of squared exponential covariance functions. In *2014 IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, 2014.

[33] David Higdon. A process-convolution approach to modelling temperatures in the north atlantic ocean. *Environmental and Ecological Statistics*, 5(2):173–190, 1998.

[34] C Paciorek and M Schervish. Nonstationary covariance functions for gaussian process regression. *Advances in neural information processing systems*, 16:273–280, 2004.

[35] Amarjeet Singh, Fabio Ramos, H Durrant-Whyte, and William J Kaiser. Modeling and decision making in spatio-temporal processes for environmental surveillance. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 5490–5497. IEEE, 2010.

[36] Alexandra M Schmidt and Anthony O'Hagan. Bayesian inference for non-stationary spatial covariance structure via spatial deformations. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 65(3):743–758, 2003.

[37] Tobias Pfingsten, Malte Kuss, and Carl Edward Rasmussen. Nonstationary gaussian process regression using a latent extension of the input space. *URL http://www. kyb. mpg. de/˜ tpfingst*, 2006.

[38] Joonki Noh and Victor Solo. Testing for space-time separability in functional mri. In *2007 4th IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, pages 412–415. IEEE, 2007.

[39] Joonki Noh and Victor Solo. Space-time separability in fmri: Asymptotic power analysis and cramér-rao lower bounds. *IEEE Transactions on Signal Processing*, 61(1):148–153, 2012.

[40] Andrea Carron, Marco Todescato, Ruggero Carli, Luca Schenato, and Gianluigi Pillonetto. Machine learning meets kalman filtering. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 4594–4599. IEEE, 2016.

[41] Jouni Hartikainen and Simo Särkkä. Kalman filtering and smoothing solutions to temporal gaussian process regression models. In *2010 IEEE International Workshop on Machine*

Learning for Signal Processing, pages 379–384. IEEE, 2010.

[42] Simo Sarkka, Arno Solin, and Jouni Hartikainen. Spatiotemporal learning via infinite-dimensional bayesian filtering and smoothing: A look at gaussian process regression through kalman filtering. *IEEE Signal Processing Magazine*, 30(4):51–61, 2013.

[43] Robert N Miller. Toward the application of the kalman filter to regional open ocean modeling. *Journal of Physical Oceanography*, 16(1):72–86, 1986.

[44] Mehran Mesbahi and Magnus Egerstedt. *Graph Theoretic Methods in Multiagent Networks*. Princeton University Press, 2010.

[45] C. Guestrin, A. Krause, and A. Singh. Near-optimal sensor placements in gaussian processes. In *International Conference on Machine Learning*, 2005.

[46] Peter J Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics*, 656:5–28, 2010.

[47] Clarence W. Rowley Williams, Matthew O. and Ioannis G. Kevrekidis. A kernel-based method for data-driven koopman spectral analysis. *Journal of Computational Dynamics*, 2(2), 2015.

[48] Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, 2016.

[49] Haim Brezis. *Functional analysis, Sobolev spaces and partial differential equations*. Springer Science & Business Media, 2010.

[50] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *NIPS*, pages 1177–1184, 2007.

[51] Christopher Williams and Matthias Seeger. Using the Nyström method to speed up kernel machines. In *NIPS*, pages 682–688, 2001.

[52] Kemin Zhou, John C. Doyle, and Keith Glover. *Robust and Optimal Control*. Prentice Hall, Upper Saddle River, NJ, 1996.

[53] W Murray Wonham. *Linear multivariable control*. Springer, 1974.

[54] Charles A Micchelli. Interpolation of scattered data: distance matrices and conditionally positive definite functions. In *Approximation Theory and Spline Functions*, pages 143–145. Springer Netherlands, 1984.

[55] Rajeev Motwani and Prabhakar Raghavan. *Randomized algorithms*. Chapman & Hall/CRC, 2010.

[56] Lehel Csatö and Manfred Opper. Sparse on-line gaussian processes. *Neural computation*, 14(3):641–668, 2002.

[57] Igor Mezić. Analysis of fluid flows via spectral properties of the koopman operator. *Annual Review of Fluid Mechanics*, 45:357–378, 2013.

[58] Clarence W Rowley, Igor Mezić, Shervin Bagheri, Philipp Schlatter, and Dan S Henningson. Spectral analysis of nonlinear flows. *Journal of fluid mechanics*, 641:115–127, 2009.

[59] Joseph Berkovitz. Action at a distance in quantum mechanics. 2007.

[60] Lixin Li, Xingyou Zhang, and Reinhard Piltner. A spatiotemporal database for ozone in the conterminous us. In *Thirteenth International Symposium on Temporal Representation and Reasoning (TIME'06)*, pages 168–176. IEEE, 2006.

[61] D. Nordby, R. Hartzler, and K. Bradley. Biology and management of waterhemp. *Glyphosate, Weeds, and Crop Sciences, Purdue University Extension, publication GWC-13.12*, 2007.

[62] Brian J Schutte and Adam S Davis. Do common waterhemp (amaranthus rudis) seedling emergence patterns meet criteria for herbicide resistance simulation modeling? *Weed Technology*, 28(2):408–417, 2014.

[63] Brent A Sellers, Reid J Smeda, William G Johnson, J Andrew Kendig, and Mark R Ellersieck. Comparative growth of six amaranthus species in Missouri. *Weed Science*, 51(3):329–333, 2003.

[64] Michael J Horak and Thomas M Loughin. Growth analysis of four amaranthus species. *Weed Science*, 48(3):347–355, 2000.

[65] Dawit Mulugeta and David E Stoltenberg. Seed bank characterization and emergence of a weed community in a moldboard plow system. *Weed Science*, pages 54–60, 1997.

[66] Dawit Mulugeta and Chris M Boerboom. Seasonal abundance and spatial pattern of setaria faberi, chenopodium album, and abutilon theophrasti in reduced-tillage soybeans. *Weed science*, pages 95–106, 1999.

[67] Adam S Davis, Sharon Clay, John Cardina, Anita Dille, Frank Forcella, John Lindquist, and Christy Sprague. Seed burial physical environment explains departures from regional hydrothermal model of giant ragweed (ambrosia trifida) seedling emergence in us midwest. *Weed science*, 61(3):415–421, 2013.

[68] Rodrigo Werle, Lowell D Sandell, Douglas D Buhler, Robert G Hartzler, and John L Lindquist. Predicting emergence of 23 summer annual weed species. *Weed Science*, 2014.

[69] B Chopard and M Droz. *Cellular automata*. Springer, 1998.

[70] Lehel Csato and Manfred Opper. Sparse representation for gaussian process models. *Advances in neural information processing systems*, pages 444–450, 2001.

[71] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497, 2015.

[72] Timothy J Richards. Immigration reform and farm labor markets. *American Journal of Agricultural Economics*, 100(4):1050–1071, 2018.

[73] Tom Hertz and Steven Zahniser. Is there a farm labor shortage? *American Journal of Agricultural Economics*, 95(2):476–481, 2013.

[74] Julie Guthman. Paradoxes of the border: labor shortages and farmworker minor agency in reworking california,Äôs strawberry fields. *Economic Geography*, 93(1):24–43, 2017.

[75] C. Morris. California crops rot as immigration crackdown creates farmworker shortage. *Fortune*, pages Online: http://fortune.com/2017/08/08/immigration–worker–shortage–rotting–crops/, 2017.

[76] Sarah Taylor Lovell, Christian Dupraz, Michael Gold, Shibu Jose, Ronald Revord, Erik Stanek, and Kevin J Wolz. Temperate agroforestry research: considering multifunctional woody polycultures and the design of long-term field trials. *Agroforestry Systems*, pages 1–19, 2017.

[77] D. Mitchell. Labor shortage provides obstacle for berry growers. *The Packer*, July 26, 2017. 2017.

[78] G. Mohan. As california's labor shortage grows, farmers race to replace workers with robots. *Los Angeles Times*, July 21, 2017 2017.

[79] Erkan Kayacan, Zhongzhong Zhang, and Girish Chowdhary. Embedded high precision control and corn stand counting algorithms for an ultra-compact 3d printed field robot. *Proceedings of Robotics: Science and Systems. Pittsburgh, Pennsylvania*, 2018.

[80] Tim Mueller-Sim, Merritt Jenkins, Justin Abel, and George Kantor. The robotanist: a ground-based agricultural robot for high-throughput crop phenotyping. In *IEEE International Conference on Robotics and Automation (ICRA), Singapore, Singapore*, 2017.

[81] Nicolas Virlet, Kasra Sabermanesh, Pouria Sadeghi-Tehran, and Malcolm J Hawkesford. Field scanalyzer: An automated robotic field phenotyping platform for detailed crop monitoring. *Functional Plant Biology*, 44(1):143–153, 2017.

[82] Søren Marcus Pedersen, S Fountas, Henrik Have, and BS Blackmore. Agricultural robots—system analysis and economic feasibility. *Precision agriculture*, 7(4):295–308, 2006.

[83] Xin-Guang Zhu, Jonathan P Lynch, David S LeBauer, Andrew J Millar, Mark Stitt, and Stephen P Long. Plants in silico: why, why now and what?—an integrative platform for plant systems biology research. *Plant, cell & environment*, 39(5):1049–1057, 2016.

[84] Elke Stehfest, Maik Heistermann, Joerg A. Priess, Dennis S. Ojima, and Joseph Alcamo. Simulation of global crop production with the ecosystem model DayCent. *Ecological Modelling*, 209(2–4):203–219, December 2007.

[85] Jonathan A. Foley, I. Colin Prentice, Navin Ramankutty, Samuel Levis, David Pollard, Steven Sitch, and Alex Haxeltine. An integrated biosphere model of land surface processes, terrestrial carbon balance, and vegetation dynamics. *Global Biogeochemical Cycles*, 10(4):603–628, December 1996.

[86] Christopher J. Kucharik. Evaluation of a process-based agro-ecosystem model (agro-IBIS) across the U.S. corn belt: Simulations of the interannual variability in maize yield. *Earth Interactions*, 7(14):1–33, December 2003.

[87] Mark A. Friedl, Damien Sulla-Menashe, Bin Tan, Annemarie Schneider, Navin Ramankutty, Adam Sibley, and Xiaoman Huang. MODIS collection 5 global land cover: Algorithm refinements and characterization of new datasets. *Remote Sensing of Environment*, 114(1):168–182, 2010.

[88] M. A. Rodrigues, D. M. Lopes, M. S. Leite, and V. M. Tabuada. Calibration and application of FOREST-BGC in NorthWestern of portugal. In *EGU General Assembly Conference Abstracts*, volume 12, page 7270, 2010.

[89] L. Nunes, M. A. Rodrigues, and D. Lopes. Evaluation of the climate change impact on the productivity of portuguese pine ecosystems using the forest-BGC model. In *Advances in Meteorology, Climatology and Atmospheric Physics*, pages 655–661. Springer, 2013.

[90] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[91] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[92] Y. Bengio. Learning deep architectures for ai. *Foundations and Trends in Machine Learning*, 2:1–127, 2009.

[93] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[94] A. Roshko. On the development of turbulent wakes from vortex streets. *California Institute of Technology*, Report 1191, 1954.

[95] M. Braza, P.H.H.M. Chassaing, and H.H. Minh. Numerical study and physical analysis of the pressure and velocity fields in the near wake of a circular cylinder. *Journal of fluid mechanics*, 165(130), 1986.

[96] B.N. Rajani, A. Kandasamy, and S. Majumdar. Numerical simulation of laminar flow past a circular cylinder. *Applied Mathematical Modelling*, 33(3):1228–1247, 2009.

[97] Graham Burnett. The seven layers of the forest garden. https://en.wikipedia.org/?title=Forest_gardening#/media/File:Forgard2-003.gif. Accessed: Dec 2018.

[98] Christopher J Rhodes. Feeding and healing the world: through regenerative agriculture and permaculture. *Science progress*, 95(4):345–446, 2012.

TABLE 1: Total training and prediction times for Figs. 10 and 11

	Intel Berkeley	**Irish** Wind	**Ozone**
Data Size (*bases-timesteps*)	25-72	12-36	30-68
Kernel Observer	2.1 sec	0.1 sec	1.61 sec
PCLSK	121.4 sec	7.0 sec	91.90 sec
LEIS	43.8 sec	2.8 sec	37.41 sec

TABLE 2: Seed Bank Density Parameters

Parameter	GCC	S_0 (seeds/cell)	Num. of Patches	Patch Size (Cells in X and Y)
Range	[0.31,0.35]	[600,1560]	50	[0,20]

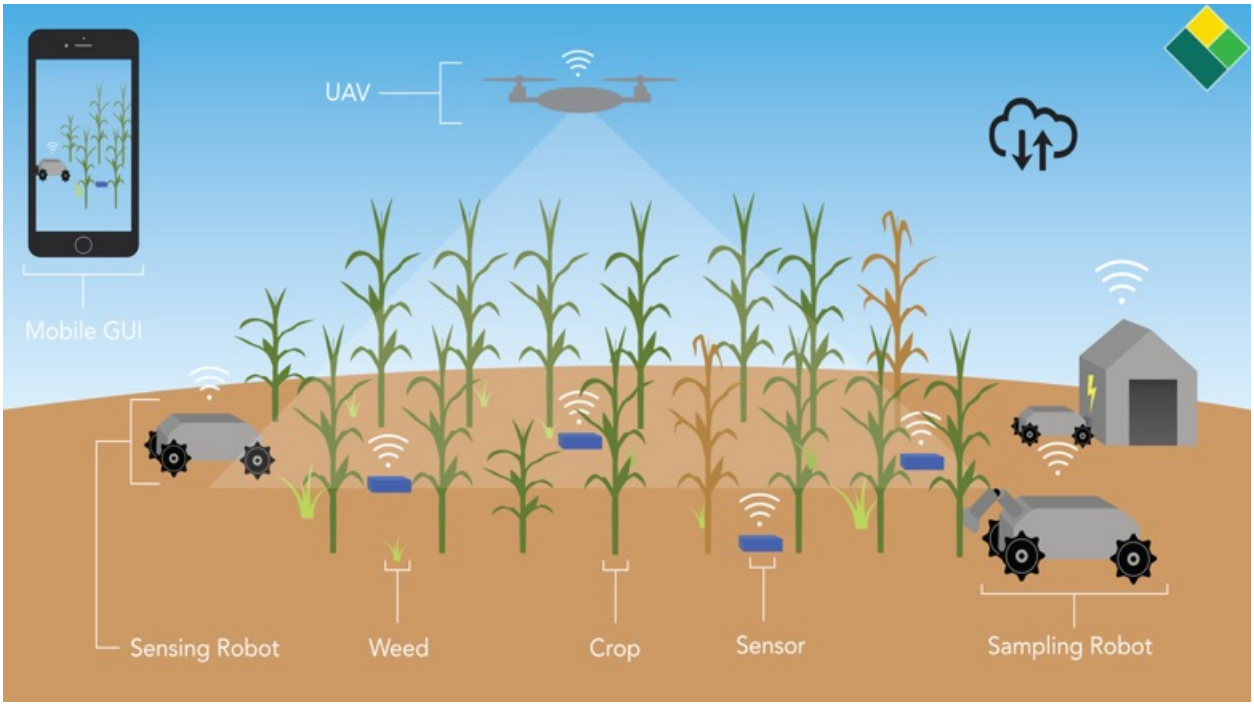


Figure 1: A Cyber Physical system consisting of a distributed team of robots for mechanical weed management on a farm. Image courtesy EarthSense inc.

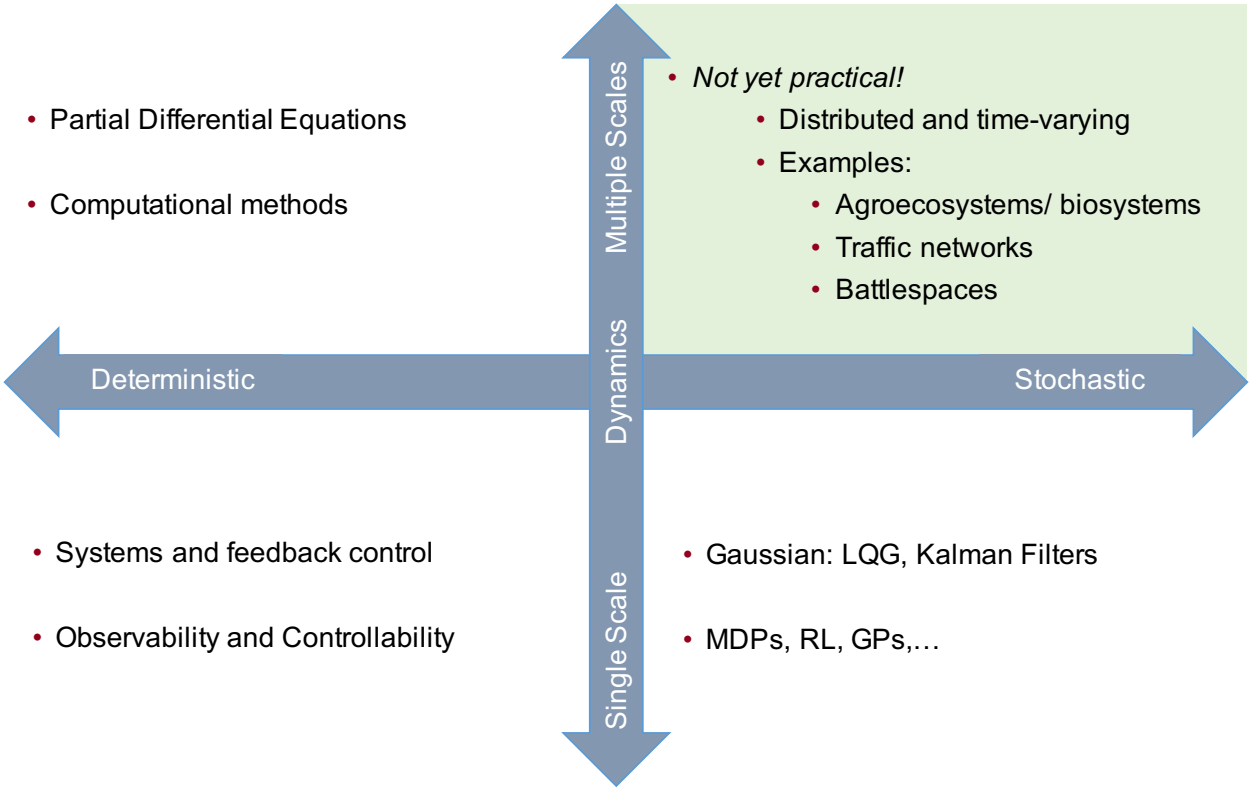


Figure 2: Modeling, monitoring, and controlling dynamical systems with complex and uncertain dynamics, such as agricultural, traffic, or weather monitoring systems, presents exciting open challenge for the controls community. The bottom left quadrant describes linear and time-invariant systems with single-scale dynamics, for which the theory of feedback control of dynamical systems is often sufficient. The bottom right quadrant shows stochastic single-scaled systems, where approaches such as Kalman Filters and Gaussian optimization have marked several successes of control systems, enabling endeavors from lunar landings to GPS navigation. The top left quadrant denotes systems with dynamics at multiple scales, where the efficient computation of solutions to Partial Differential Equations (PDEs) is an highly active area of research. Fundamental theoretical advances and practical algorithms are needed, however, to enable autonomous decision making for distributed stochastic Cyber Physical Systems with dynamics at multiple scales – shown in the top right quadrant – such as distributed agricultural robotic systems, traffic networks, and weather monitoring systems with mobile and stationary sensors.

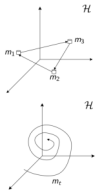
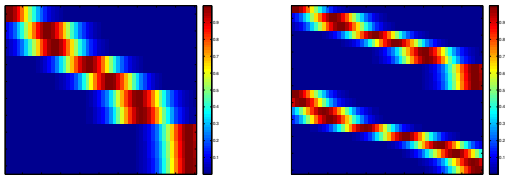
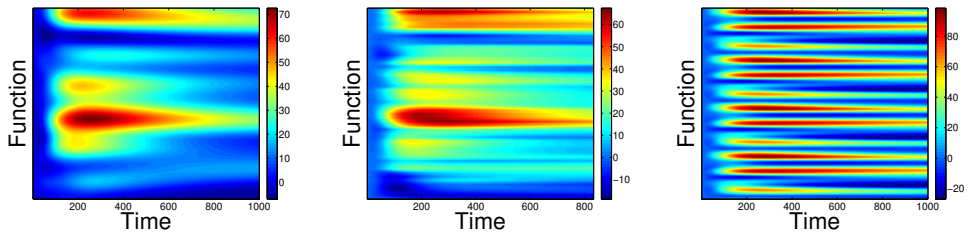


Figure 3: Two types of Hilbert space evolutions. Left: discrete switches in RKHS $\mathcal{H}$; Right: smooth evolution in $\mathcal{H}$.



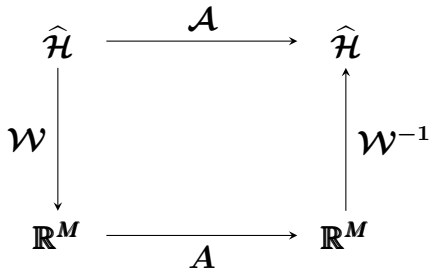
(a) 1-shaded (Def. 1) (b) 2-shaded (Eq. (8))

Figure 4: Shaded observation matrices for dictionary of atoms. Each row represents a sensing location with the color map indicating the evaluation of kernel function w.r.t the others points in the domain.

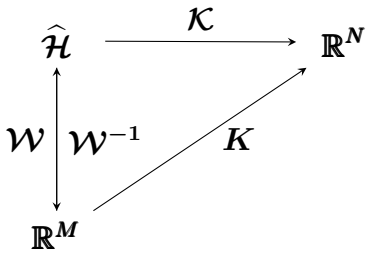


(a) Gaussian (b) Laplacian (c) Periodic

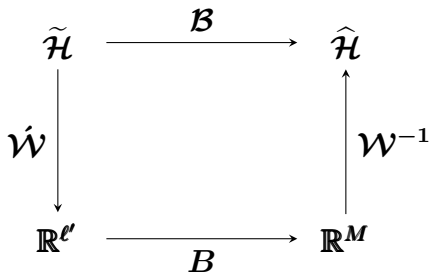
Figure 5: One-dimensional function evolution over a fixed transition matrix A, initial condition w_0 and centers $\mathcal{C}$, but with different kernels $k(x, y)$. Each y-vector at a given value of x represents the output of the function, which evolves from left to right. As seen, changing the kernel creates quite different dynamic behaviors.



(a) Relationship between $\mathcal{A}$ and A



(b) Relationship between $\mathcal{K}$ and K



(c) Relationship between $\mathcal{B}$ and B

Figure 6: Commutative diagrams between primal and dual spaces

Algorithm 1 Measurement Map $\widetilde{K}$

Input: $\widehat{A} \in \mathbb{R}^{M \times M}$
Compute Rational Canonical Form, s.t. $C = Q^{-1}\widehat{A}^T Q$. Set $C_0 := C$, and $M_0 := M$.
for $i = 1$ **to** ℓ **do**
  Obtain (m.p.) $\alpha_i(\lambda)$ of C_{i-1}. This returns associated indices $\mathcal{J}^{(i)} \subset \{1, 2, \ldots, M_{i-1}\}$.
  Construct vector $v_i \in \mathbb{R}^M$ such that $\xi_{v_i}(\lambda) = \alpha_i(\lambda)$.
  Use indices $\{1, 2, \ldots, M_{i-1}\} \setminus \mathcal{J}^{(i)}$ to select matrix C_i. Set $M_i := |\{1, 2, \ldots, M_{i-1}\} \setminus \mathcal{J}^{(i)}|$
end for
Compute $\mathring{K} = [v_1^T, v_2^T, ..., v_\ell^T]^T$
Output: $\widetilde{K} = \mathring{K} Q^{-1}$

Algorithm 2 Sampling locations set $\mathcal{X}$

Input: $\widehat{A} = C$, lower bound ℓ

Decompose C to generate invariant subspaces $\widehat{\mathcal{H}}_j$, $j \in \{1, 2, \ldots, \ell\}$ (see section "Preliminaries on Rational Canonical Structures")

for $j = 1$ **to** ℓ **do**

Obtain centers $\mathcal{C}^{(j)}$ w.r.t subspace $\widehat{\mathcal{H}}_j$,

Generate samples $x_i^{(j)}$ to create a kernel matrix $K^{(j)}$ that is shaded only with respect to centers $\mathcal{C}^{(j)}$

end for

Output: Sampling locations set $\mathcal{X} = \{x^{(1)}, x^{(2)} \cdots, x^{(l)}\}$.

Algorithm 3 Kernel Observer (Transition Learning)

Input: Kernel k, basis centers $\mathcal{C}$, final time step T.
while $\tau \leq T$ **do**
1) Sample data $\{y_\tau^i\}_{i=1}^M$ from f_τ.
2) Solve for $\widehat{w}_\tau$ by using least squares on $y_\tau = K\widehat{w}_\tau$.
3) Store weights $\widehat{w}_\tau$ in matrix $\mathcal{W} \in \mathbb{R}^{M\times T}$.
end while
To infer $\widehat{A}$, define matrix $\Phi = \mathcal{W}^T\mathcal{W}$. Then:
for $i = 1$ **to** M **do**
At step i, solve system

$$\widehat{A}^{(i)} = \left((\Phi + \lambda I)^{-1}\left(\mathcal{W}^T\mathcal{W}^{(i)}\right)\right)^T, \tag{17}$$

where $\widehat{A}^{(i)}$, and $\mathcal{W}^{(i)}$ are the ith columns of $\widehat{A}$ and $\mathcal{W}^{(i)}$ respectively.
end for
Compute the covariance matrix $\widehat{B}$ of the observed weights $\mathcal{W}$.
Output: estimated transition matrix $\widehat{A}$, predictive covariance matrix $\widehat{B}$.

Algorithm 4 Kernel Observer (Monitoring and Prediction)

Input: Kernel k, basis centers $\mathcal{C}$, estimated system matrix $\widehat{A}$, estimated covariance matrix $\widehat{B}$.
Compute Observation Matrix: Compute the cyclic index ℓ of $\widehat{A}$, and compute K.
Initialize Observer: Use $\widehat{A}$, $\widehat{B}$, and K to initialize a state-observer (such as Kalman filter (KF)) on $\widehat{\mathcal{H}}$.
while measurements available **do**
1) Sample data $\{y_\tau^i\}_{i=1}^N$ from f_τ.
2) Propagate KF estimate $\widehat{w}_\tau$ forward to time $\tau+1$, correct using measurement feedback with $\{y_{\tau+1}^i\}_{i=1}^N$.
3) Output predicted function $\widehat{f}_{\tau+1}$ of KF.
end while

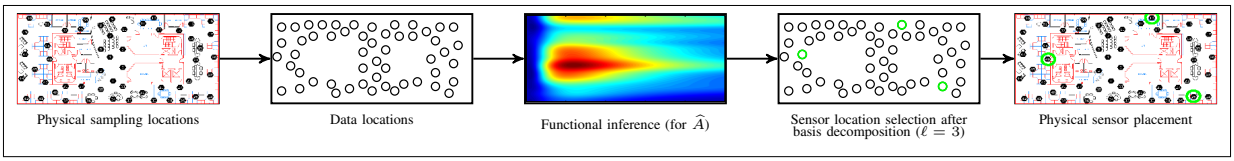


Figure 7: Overall description of how the kernel observer fits in the sensing framework. Physical locations are mapped to data locations, over which historical data is collected as a time series. Functional inference is performed over $\widehat{\mathcal{H}}$ to solve for $\widehat{A}$. The measurement operator K is then computed (see Figure 8), leading to sensor placement.

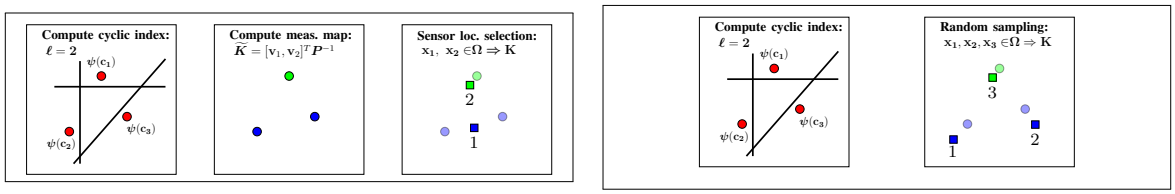
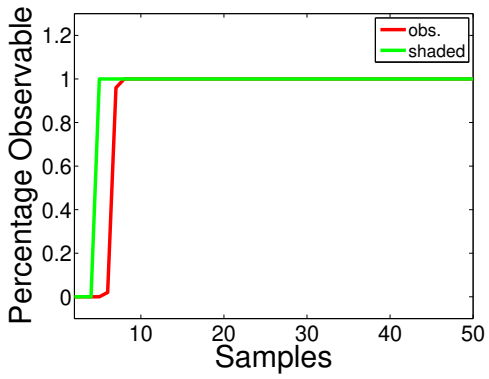


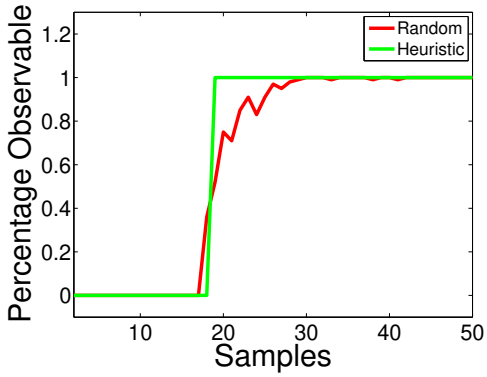
Figure 8: Diagram demonstrating sensor placement using the measurement map or random sampling approaches. The circles represent data locations associated to bases (such as $c_j \Leftrightarrow \psi(c_j)$) and the squares represent sensor locations (such as $x_i \Leftrightarrow \psi(x_i)$) . The cyclic index ($\ell = 2$) indicates how many possible couplings of bases exist, which can be represented as a choice of $\binom{M}{\ell}$ hyperplanes in Ω. If the measurement map is computed (left), the correct couplings are chosen (green vs. blue), and a smaller number of sensors (2) can be placed. Alternatively, random sampling (right) is more computationally efficient, but generally requires more sensors (3).

Algorithm 5 k-Invariant Subspaces Algorithm

while clusters have changed **do**
  for each center i **do**
    Find cluster k which maximizes the score $\frac{1}{|S_k|}\left(\|\bar{A}_{i,S_k\setminus\{i\}}\|^2 + \|\bar{A}_{S_k\setminus\{i\},i}\|^2\right)$
    Reassign center i to cluster with highest score
  end for
end while
return clusters

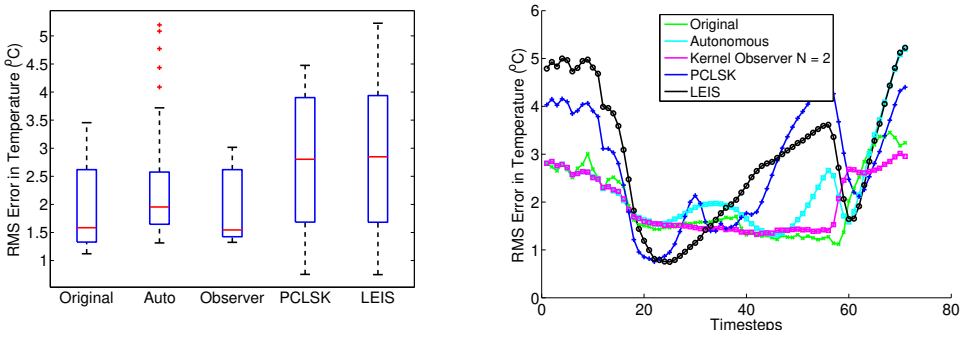


(a) Shaded vs. observability



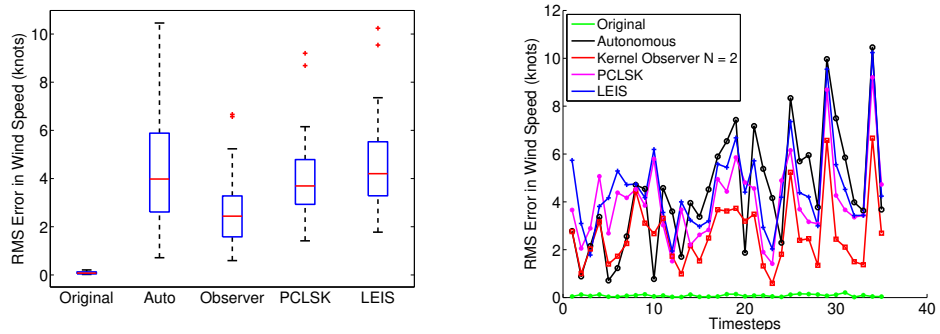
(b) Heuristic vs. random

Figure 9: Kernel observability results.



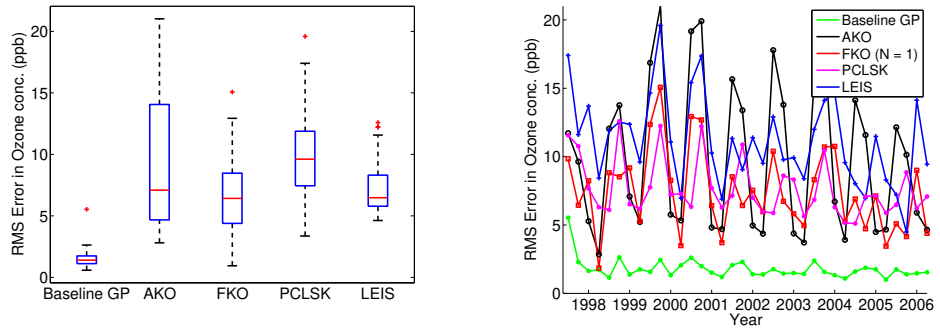
(a) Error (boxplot) (b) Error (time-series)

Figure 10: Comparison of kernel observer to PCLSK and LEIS methods on Intel Berkeley dataset.



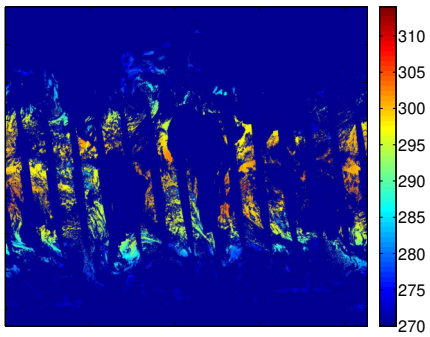
(a) Error (boxplot) (b) Error (time-series)

Figure 11: Comparison of kernel observer to PCLSK and LEIS methods on Irish Wind dataset.

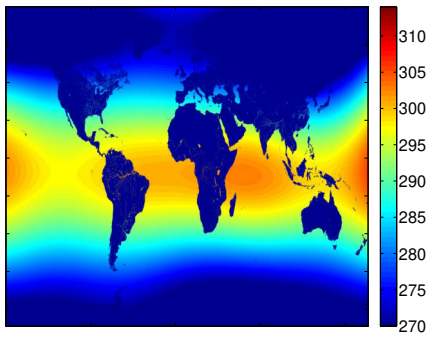


(a) Error (boxplot) (b) Error (time-series)

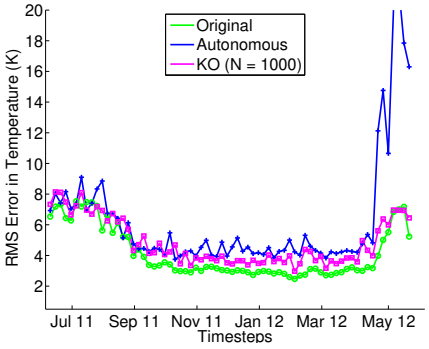
Figure 12: Comparison of kernel observer to PCLSK and LEIS methods on Ozone dataset.



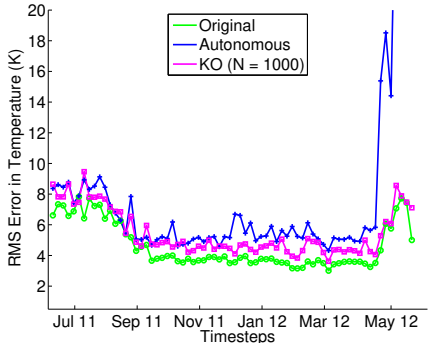
(a) Raw Satellite Data



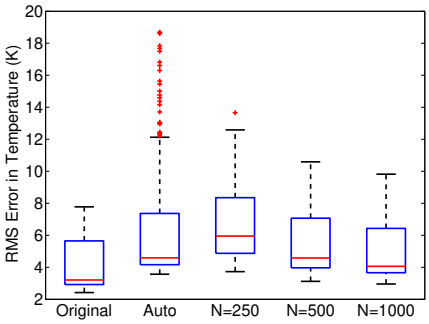
(b) AKO estimate



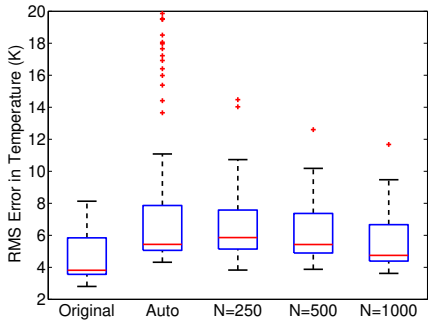
(c) Error-day (time-series)



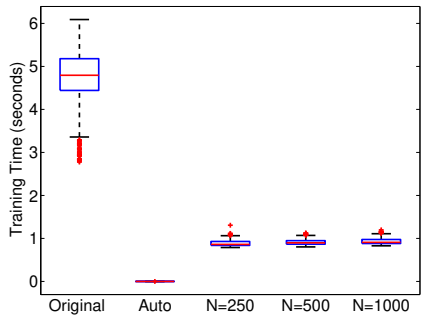
(d) Error-night (time-series)



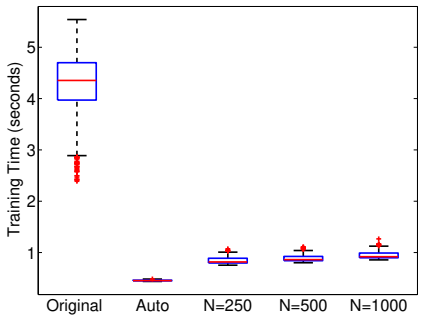
(e) Error-day (box-plot)



(f) Error-night (box-plot)

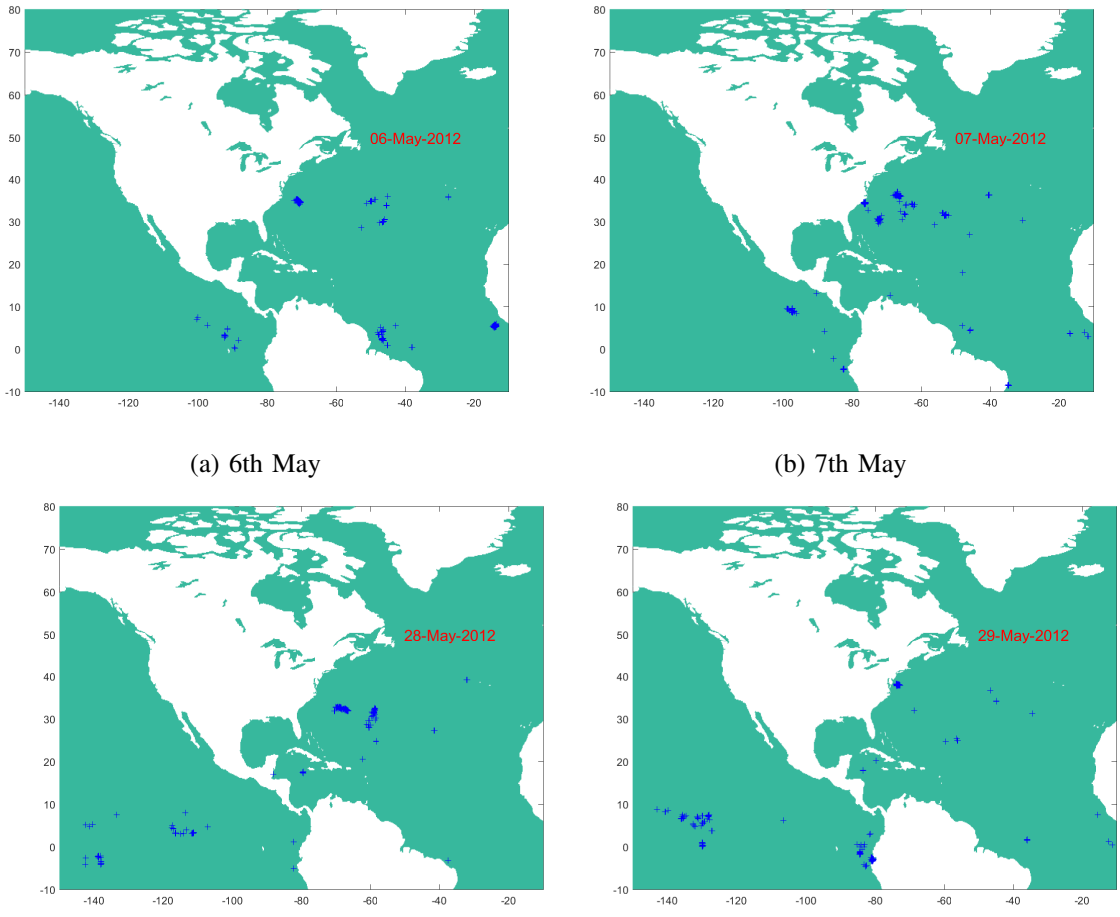


(g) Estimation time (day)



(h) Estimation time (night)

Figure 13: Performance of the kernel observer over AVVHR satellite 2012 data with different numbers of observation locations.



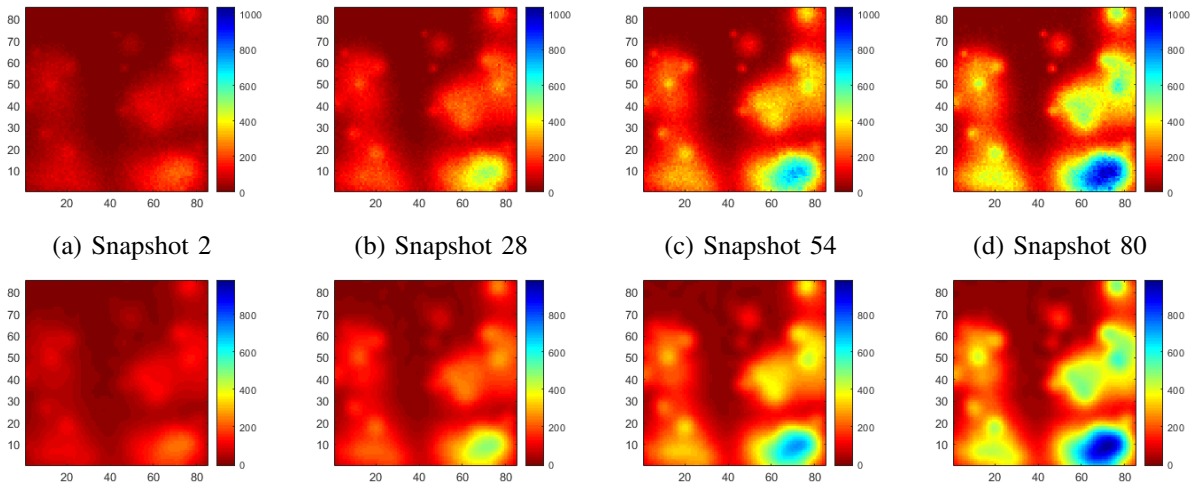
(a) 6th May

(b) 7th May

(c) 28th May

(d) 29th May

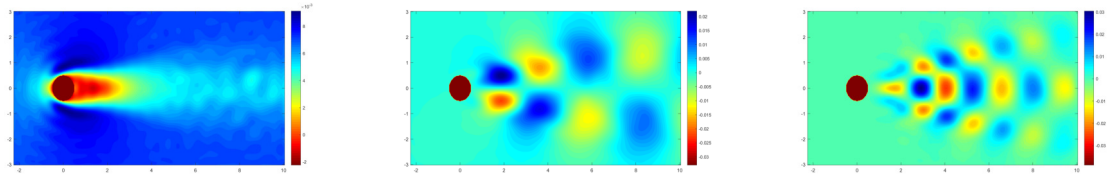
Figure 14: Locations of weather anomaly obtained based on the error between the acutual temperature and the prediction of autonomous kernel observer. Landmass is shown in white and the ocean is in green. Locations marked have error greater than two standard deviations above the mean error.



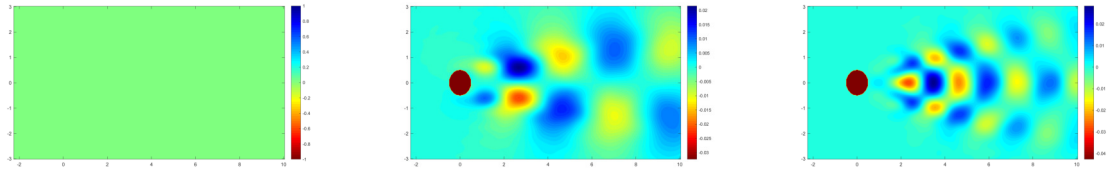
(a) Snapshot 2 (b) Snapshot 28 (c) Snapshot 54 (d) Snapshot 80

(e) Snapshot 2 (f) Snapshot 28 (g) Snapshot 54 (h) Snapshot 80

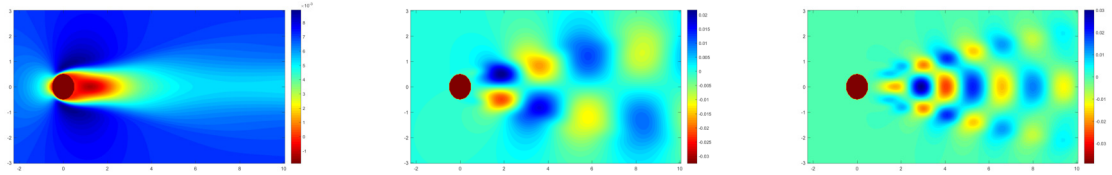
Figure 15: Visualization of Weed Density Growth over 20 days, original (a-d), E-GP (e-h)



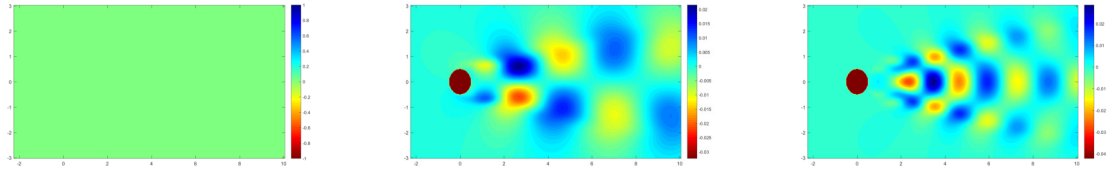
(a) 0°, real component (b) 6°, real component (c) 12°, real component



(d) 0°, imaginary component (e) 6°, imaginary component (f) 12°, imaginary component

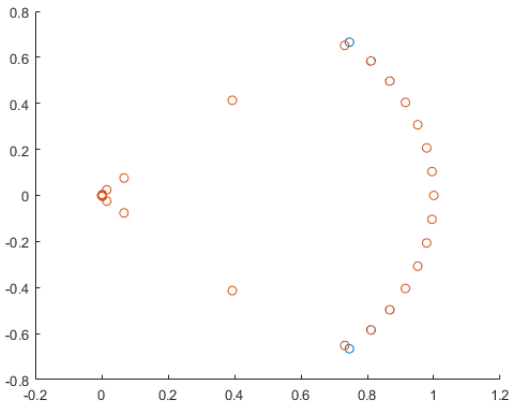


(g) 0°, real component (h) 6°, real component (i) 12°, real component

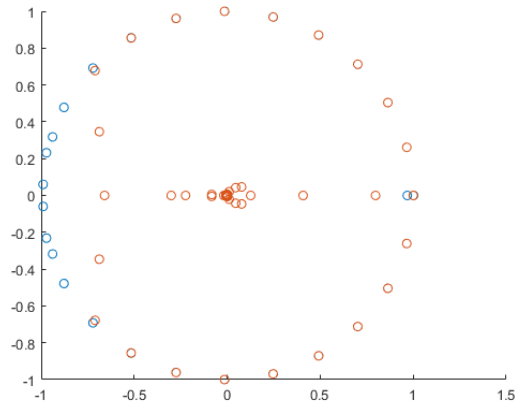


(j) 0°, imaginary component (k) 6°, imaginary component (l) 12°, imaginary component

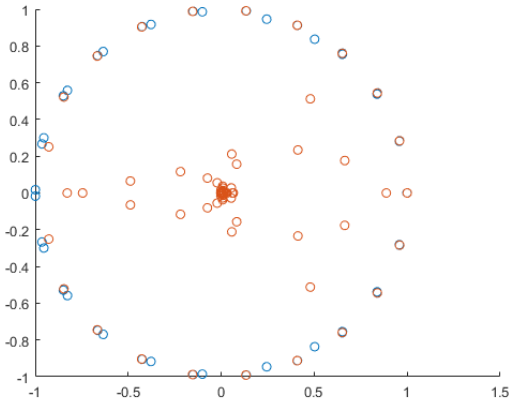
Figure 16: Primary Koopman Modes of a flow past a cylinder at Reynolds number 100. (a-f) E-GP, (g-l) DMD



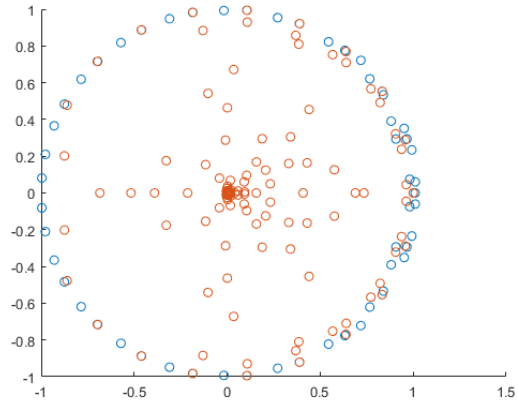
(a) Re=100



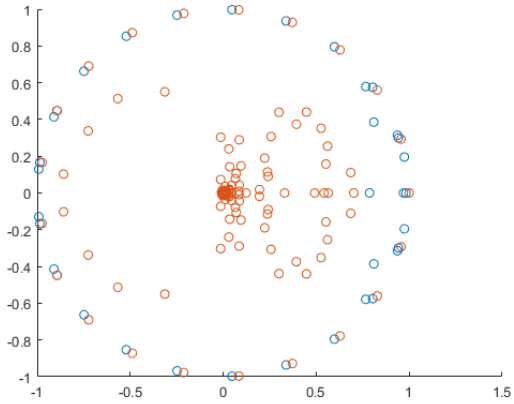
(b) Re=300



(c) Re=600



(d) Re=800



(e) Re=1000

Figure 17: Eigenvalue Comparison between E-GP and DMD at different Reynolds numbers

Sidebar: Summary

This article should be useful for anyone interested using robots in large-scale environments that are changing in time and space. We have used the methods presented here to help teams of robots monitor and destroy weeds within a field of crops. In general, this article is about modeling and monitoring complex systems that vary in both space and time, given a limited number of agents or sensors providing measurements spread out over a large area. We present a novel method for solving this problem, with several tremendously useful properties. First, it can be easily trained and updated even with large, "dirty" data collected at many places at many times. Secondly, this model lends itself well to the kinds of analysis familiar to the controls community, which means that several formerly very challenging problems become much easier: how to predict future evolution, deciding how many sensors are needed and where to place them, and what are the basic structures underneath the system dynamics. As far as we know, the methods presented here are unmatched in their scope and power. A graduate-level mathematical background is recommended for this paper, and an open code repository is provided for ease of implementation.

Sidebar: Key control problems in agriculture

Shortage of qualified human labor is a key challenge facing farmers [72], [73], leading to smaller profit margins, and preventing the adoption of truly sustainable agricultural practices. Lack of timely available labor was a principle reason behind the tens of millions of dollars of unharvested fruits and vegetables that rotted in California farms in 2017 [74], [75]. Labor shortages can be a major barrier to more sustainable agricultural practices that are more labor intensive.

For example, more sustainable alternatives to prevalent methods of agriculture that would not need large amounts of chemicals and other inputs, such as perennial polycultures (mixed species of fruit- and nut-producing trees and shrubs [76], see Figure S2), are currently impractical at scale with current agricultural equipment. Polyculture systems can leverage co-habitation of mutually beneficial plants (and animals, insects, or microbiomes) to create a more sustainable engineered ecosystem. Labor shortage has become a primary barrier to adoption of this sustainable agricultural alternative [77], [78].

One way to address the challenges of labor shortages in agriculture is by creating new robotic technology that can work in harsh, uncertain, and dynamically changing field environments. Here we outline some of the fundamental challenges in autonomy, estimation, and control that the controls community can help overcome to enable the future of agricultural robotics, and relate it with the problem of spatiotemporal function estimation studied in this paper:

Persistent multi-agent autonomy under partial observability: The digital farm of the future will employ teams of distributed heterogeneous agents to autonomously manage, optimize, and harvest large acres of diverse crops across the entire season without encumbering humans. This level of autonomy in unstructured field environments is out of reach of the current state-of-the-art which requires constant human monitoring and oversight, especially in the presence of change or unforeseen events. Efficient and reliable control will be central to the success of these robots. Small below-canopy robots (such as Figure S1 [79]) will need to provide precision care including pruning, weeding, and re-seeding without damaging plants or causing soil compaction. Deployed at scale, these robots can not only make large scale organic farming practical, but also enable enhanced breeding through field-scale phenotyping [79]–[81]. *The big controls challenge here is in making decisions over large spatiotemporal scales with information obtained from a few stationary and mobile sensors which can only partially observe the environment at any given time*. The work pursued in this paper lays a foundation towards this problem by enabling a team of agents to estimate the varying state of the environment.

Dexterous and ubiquitous robotics for precise care : The digital farm of the future will strive to eliminate costly inputs (chemicals, labor, energy, and knowledge) with low-cost, dexterous, and highly autonomous agricultural equipment [82]. Advances in *soft* arms and grippers can enable robots that can have far better reach and dexterity around plants than robots equipped with traditional *hard* industrial robotic arms. Soft arms, which are often actuated with pressurized tubes, can be far less expensive to manufacture and significantly lighter than their hard counterparts. On the other hand, soft arms can be slow to actuate and have limited payloads. To make soft robots practical, optimal feedback control techniques are necessary that work with conformal objects with very large degrees of freedom. In particular, soft arms tend to significantly deform under weight and behave quite differently when loaded with different payloads. Unlike hard arms, encoders are not sufficient to estimate the pose of the arm or the manipulator. Strain and angle sensors need to be positioned judicially to keep costs down, and image based feedback control will be necessary. The evolving Gaussian Process technique and the kernel observer techniques described in this paper could be utilized to create distributed observers for such soft systems.

The complex interaction between closely-spaced diverse plant species in a polyculture results in both spatial and temporal dynamics as the plants grow and interact with each other. Plant growth often exhibits hybrid dynamical systems behavior with rapid thresholded growth bursts followed by slow progression. The triggers for growth bursts are dependent on environmental factors such as temperature, soil moisture, and sunlight reaching individual and cumulative thresholds, and complex interrelations between neighboring plants, soil chemistry, insects, and soil microbes.

Simulating individual plant growth is an active area of research with many open questions in modeling and plant biology [83]. Arguably, very high resolution plant growth models may not even be necessary for effective control of polycultures. Yet on the other hand, the existing models of plants and interactions with ecosystems [84]–[89] are not well suited for designing and managing polycultures because of the simplifying assumptions that are often made. A good balance could be struck with data driven machine learning models that have sufficient resolution for aggregate prediction over multiple spatiotemporal scales and are lightweight enough for control and decision making. With these models, predictive control strategies can be created that task teams of robots for management tasks. Furthermore, these predictive models can enable quantified mechanisms of design and planning of efficient agroecosystems. Obtaining the required data to train these models and using them to create effective control techniques for managing profitable polycultures remains an exciting direction of future work where the controls community can help.

The discussion in this sidebar was informed by many conversations with crop scientists at UIUC. In particular, *the authors wish to thank Prof. Stephen Long and Prof. Carl Bernacchi for insights on the phenotyping bottleneck and simulation of plants (crops-in-silico), Prof. Adam Davis on inputs on the herbicide resistant weed crisis, and Prof. Sarah Lovell on sustainable agricultural production systems and perennial polycultures, as well as the members of the UIUC Center for Digital Agriculture*

Sidebar: Feature Spaces in Machine Learning

Suppose we have data $\mathcal{D} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$, where $x_i \in \Omega_I$ and $y_i \in \Omega_O$. Here, Ω_I is the input domain, and is generally a subset of $\mathbb{R}^D$, although more general sets such as discrete spaces, graphs, or text documents can be considered. Similarly, Ω_O, the output domain, can be just as general as Ω_I. We wish to solve for functions f in some space of functions $\mathcal{J}$ such that $f(x_i) = y_i \ \forall i$. Generally, to restrict the complexity of the space $\mathcal{J}$, a *loss function* $L(f, \mathcal{D}) \mapsto \mathbb{R}$ is chosen, which measures the error between a prediction $f(x_i)$ given a datapoint x_i, and y_i, averaged over the entire dataset $\mathcal{D}$, and the optimization problem becomes

$$f^* = \arg\min_{\mathcal{J}} L(f, \mathcal{D}) + \lambda g(f), \tag{S1}$$

where $\lambda \in \mathbb{R}$, and $g(f)$ represent some constraints on the function f, such as smoothness. Control theorists are most likely familiar with input-output pairs where $x_i \in \mathbb{R}^N$ and y_i is either in $\mathbb{R}$ or $\mathbb{R}^M$ (*regression*). In machine learning, the most common task is when the y_i are discrete (*classification*). Different combinations of task, loss functions, and spaces $\mathcal{J}$ result in different algorithms to solve these problems, which can sometimes form entire subfields of machine learning.

The choice of the function space $\mathcal{J}$ can be critical for the task we want to perform, similar to how the choice of the state space is in control theory. Let's consider a simple example. Suppose we have data from two *classes* $\mathcal{D}_A = \{(x_1^A, y_1^A), \ldots, (x_N^A, y_N^A)\}$ and $\mathcal{D}_B = \{(x_1^B, y_1^B), \ldots, (x_N^B, y_N^B)\}$, where $x_i^{\{A,B\}} \in \mathbb{R}^D$, and $y_i^{\{A,B\}} \in \{-1, +1\}$, shown in Figure S3. Let f be chosen from the class of linear algorithms, i.e. $f = w^T x + b$, where $w \in \mathbb{R}^D, b \in \mathbb{R}$. We pick a loss L that returns a loss of zero when the prediction is the correct class, and if the prediction is the incorrect class, returns a higher value for misclassifications that are closer to the boundary. A classical example of such a loss is that used by the *perceptron algorithm*, which can be written as

$$L(f, \mathcal{D}) = \frac{1}{N} \sum_{i=1}^{N} \max(0, -y_i w^T x). \tag{S2}$$

This loss measures how accurate the prediction of the perceptron is on average. The general algorithm is as follows:

1) Initialize $w \in \mathbb{R}^D$ to all zeros.
2) For a fixed number of iterations, or until some stopping criterion is met:
    a) For each training example (x_i, y_i),
        i) Let $\hat{y}_i = \text{sgn}(w^T x_i)$.
        ii) If $y_i \neq \hat{y}_i$, update $w \leftarrow w + y_i x_i$.

The perceptron was one of the first machine learning models, and the genesis of modern neural networks [90]. Figure S3 shows a visual representation of where the perceptron algorithm can solve for the decision boundary with zero error. However, if the structure of the data has some nonlinearities, no solution will be found, as seen in Figure S4. In this case, the original space the data resides in is, in some sense, not a rich enough representation. If we could construct a mapping of the data to a different space which gives a learning algorithm more degrees of freedom to work with, linear algorithms can still be deployed. If we map the same data using a nonlinear map $\phi(x, y) := (x^2, y^2, 2xy)$, the perceptron now finds a solution in 3 dimensions, as seen in Figure S5. This example shows why so much of the work in machine learning focuses on learning the right representation for the data, for the right representation makes the classification task easy. Two major threads of research in the arena of feature maps over the last 40 years are kernel methods, and neural networks, the latter of which has gained remarkable notoriety in the last 10 years. These lines of research represent distinctly different strategies for generating feature maps from data.

In Figure S4, the data was mapped using an explicit feature map. Kernel methods, of which Gaussian Processes are a great example, utilize an elegant strategy for generating feature maps from data, using a remarkably simple trick called *the kernel trick*. Given a positive-definite kernel function $k(x, y) : \Omega \times \Omega \to \mathbb{R}$, Mercer's theorem guarantees the existence of a feature map $\psi : \Omega \to \mathcal{H}$, where $\mathcal{H}$ is a *reproducing kernel Hilbert space (RKHS)*, and the map ψ obeys the property

$$k(x, y) = \langle \psi(x), \psi(y) \rangle_{\mathcal{H}}. \tag{S3}$$

Recall that since $\mathcal{H}$ is an RKHS, given $c \in \Omega$, $k(x, c) = \langle \psi(x), \psi(c) \rangle_{\mathcal{H}}$, and $k(x, c) := \psi_c \in \mathcal{H}$. Furthermore, $\mathrm{span}\{\psi_x\}_{x\in\Omega}$ is dense in $\mathcal{H}$. There exist kernels that generate $\mathcal{H}$s that are extremely high dimensional: for example, the RBF kernel $k(x, y) = e^{-\gamma\|x-y\|^2}$ is infinite-dimensional. This high degree of freedom enables the design of powerful learning algorithms that are linear in $\mathcal{H}$, but nonlinear in the input domain Ω. The canonical example of this is the support vector machine (SVM) [91], but a more instructive example for us is the perceptron algorithm.

Suppose we trained a percetron using $\mathcal{X} = \{x_1, \ldots, x_N\}$, $x_i \in \mathbb{R}^D$ for some D, $y_i \in \{-1, 1\}$. The prediction of the perceptron is $\hat{y} = \mathrm{sgn}(w^T x)$, where $w \in \mathbb{R}^D$. It can be shown that $w = \sum_{i=1} \alpha_i y_i x_i$, where α_i is the number of times x_i was misclassified. This allows us to derive the dual version of this algorithm, because

$$\hat{y} = \mathrm{sgn}(w^T x) = \mathrm{sgn} \sum_{i=1}^{n} \alpha_i y_i \langle x_i, x \rangle_{\mathbb{R}^D}.$$

The dot product $\langle x_i, x \rangle_{\mathbb{R}^D}$ can be replaced with the kernel, leading to the *kernel perceptron algorithm*:

1) Initialize $\alpha \in \mathbb{R}^N$ to all zeros.
2) For a fixed number of iterations, or until some stopping criterion is met:
    a) For each training example (x_j, y_j),
        i) Let $\hat{y}_i = \operatorname{sgn} \sum_{i=1}^{n} \alpha_i y_i k(x_i, x_j)$.
        ii) If $y_i \neq \hat{y}_i$, update $\alpha_j \leftarrow \alpha_j + 1$.

This algorithm is nonlinear in the input domain, but linear in the feature space $\mathcal{H}$, which led the deep learning community to, somewhat pejoratively, label this as an example of a *shallow learning architecture*. Kernel methods can also be used in a more direct fashion: if we have a subspace $\mathcal{H}' \subset \mathcal{H}$ with a basis generated from $\mathcal{C} = \{c_1, \ldots, c_M\}$, i.e. $\mathcal{H}' = \operatorname{span}\{\psi_{c_1}, \ldots, \psi(c_M)\}$, a linear model in $\mathcal{H}'$ is again given by a vector $w \in \mathbb{R}^M$. Suppose this weight vector represents a boundary in $\mathcal{H}'$: to compute which side of this boundary a point x would lie on in $\mathcal{H}'$, we simply compute $\operatorname{sgn}(\sum_{i=1}^{M} w_i \langle \psi(x), \psi(c_i) \rangle_{\mathcal{H}}) = \operatorname{sgn}(\sum_{i=1}^{M} w_i k(x, c_i))$. The choice of the kernel and its parameters depends on the dataset and the loss function. The kernel and the data together form the feature space. Because kernel methods are linear in their parameters and are restricted to RKHSs, they are amenable to somewhat straightforward mathematical analysis, and are very well studied because of this. The very recent review of Gaussian processes in control that appeared in IEEE CSM contains further details, examples and pointers to software relating to GPs and their use in control [19]. We switch our attention now to a different way of obtaining the features: Deep Neural Networks. While GPs build features using positive semidefinite symmetric kernels that compare any two points, DNNs build features using nested nonlinear operations.

Deep neural networks (DNNs) are models where the representing function f has nested nonlinearities. Fix a width $M \in \mathbb{N}$. Deep nets are parameterized models with weight matrices $W^l \in \mathbb{R}^{M \times M}$, bias vectors $b^l \in \mathbb{R}^M$, and a pointwise nonlinearity $\phi : \mathbb{R} \to \mathbb{R}$, with $l = 1, \ldots, L$. Vectors $h^l \in \mathbb{R}^M$ are called preactivations, and $x^l \in \mathbb{R}^M$ are called postactivations, each element of which is called a *neuron*. Let $h^0 \in \mathbb{R}^M$ be the input: then the canonical feedforward neural network is given by

$$x^l = \phi(h^l), \; h^l = W^l x^{l-1} + b^l.$$

Therefore, we have that $f(h^0) = x^l$. The individual steps l are called the *layers* of the network, and the nesting property allows these networks to learn much more complicated functions than shallow architectures given the same number of nonlinearities [92]. Different choices of nonlinearities, connections, and layer architectures lead to different types of neural networks, which are used for different applications [93]. Deep learning has had an enormous impact on both the machine learning literature and industrial applications, and that impact has

bled over rapidly to other fields. Due to the nested structure of nonlinearities in DNNs, they are more difficult to analyze using simple mathematical tools, and therefore most of the literature in the field has focused on the empirical performance of these methods, where they significantly outperform competing methods. The significance of the achievements of machine learning in general and deep learning in particular have been in its ability to simplify the implementation of complex functional representation, essentially, in its ability to make system identification more accessible for difficult problems. However, the feature spaces generated by deep networks are not as well behaved and accessible to analysis as those generated by kernel models. In particular, deep network feature spaces do not naturally have the properties of RKHSs. This presents one barrier to analyzing and understanding the nature of these models. For the reliable and verifiable inclusion of DNNs in engineering control systems, further insights are necessary into the structure of the feature spaces these models generate, to restrict certain forms of output, and to shape decision-making.

Sidebar: Learning Fluid Flows with Evolving Gaussian Processes

2nd order Partial Differential Equations (PDEs) are ubiquitous in practical science and engineering, from mechanics to transport phenomena to electromagnetics. In our view, the Navier-Stokes equations governing fluid dynamics represent most, if not all of the overall complexity of modeling these as (a) it exhibits of hybrid system behavior, such as elliptic-hyperbolic, and (b) the nonlinearity results in the complex spatio-temporal dynamics that are prevalent in many practical situations.

We demonstrate the Evolving Gaussian Processes method on CFD data of flow over a bluff body (a cylinder) over a range of Reynolds numbers from 100 to 1000 (the Reynold number is a dimensionless flow rate). This deterministic, high-dimensional spatiotemporal dynamical system is well-studied in the fluid dynamics literature, both experimentally and numerically [94]–[96]. The conventional wisdom would be to learn a separate model over each Reynolds number, but our results show that the E-GP method is capable of learning the dynamics of all the flow patterns at once. Using the learned dynamics over weights of successive kernel models, E-GP is capable of predicting the future states of functional evolution in a recursive manner. The key advantage of E-GP is that evolution of large function spaces can be transformed into learning the evolution in a relatively smaller Hilbert space which is encoded by the kernels and the associated weight vector.

In our CFD simulation, we used a 4th-order polynomial expansion with the spectral element method on the incompressible Navier-Stokes equation to generate the cylinder flow data for Re = 100, 300, 600, 800, and 1000. The spatial domain is $[-2, 10] \times [-3, 3]$, excluding the diameter-1 cylinder at the origin. Neumann boundary conditions are applied to the far-field of the cylinder in the y-direction and the outlet of the flow field; and a Dirichlet boundary condition is applied to the inlet. Each dataset contains at least 200 snapshots with a uniform time step of 0.03 sec. Each snapshot contains 24,000 velocity data points for Re=100 or 95,000 velocity data points for Re=300,600,800,1000. Each dataset took at least 10 hours in a high performance computer cluster to generate. Figures S6,S7(a-d) visualize the horizontal velocity for Re=100 and Re=1000, with red being the greatest negative velocity and blue the greatest positive velocity. The flow is unstable, periodic, and clearly nonlinear.

We used the Gaussian RBF kernel $k(x, y) = e^{-\|x-y\|^2/2\sigma^2}$ in our E-GP model, with σ estimated to be 0.4. Using a budget of 600 kernel centers (see Figure S9(a)-S9(b), and note how they cluster in the most dynamic regions), we find a 600×600 matrix $\widehat{A}$ which accurately (Figure S8(a)) captures the dynamics of the nonlinear system. We can use this to propagate a single initial condition w_0 forward to make predictions, then compare the predictions to the

original training data. We found total percentage errors between 3% for Re=100 and 7-8% for Re=1000, as can be seen in the solid lines in Figure S8(a). We define the total percentage errors as $E_\tau = \frac{\|y_\tau - \bar{y}_\tau\|_2}{\|\bar{y}_\tau\|_2}$ where $\bar{y}_\tau$ is the output vector for time τ and y_τ is the E-GP estimate at that time. Note that *the size of the model has been reduced by almost two orders of magnitude* from the original CFD data. This process takes about 13 minutes in MATLAB for a 200 snapshot by 95,000 point set on an ordinary Intel i7 4.00 GHz processor.

One Transition Matrix for Everything

To approach the challenge of generalizing across similar spatiotemporally evolving systems, the first question to answer is whether we can find an $\widehat{A}$ matrix that accurately captures the dynamics of multiple similar flows. The answer to that question is *yes*, using the trajectory concatenation method. Amazingly, a single model generated this way works almost as well on all five datasets as do five individual models trained on each dataset separately. This is confirmed by both the total error plots (Figure S8(a)), which show only slight increases in each of the total percentage error plots, and visual inspection of the dynamic modes displayed. This result is even more surprising in light of the fact that the rate of vortex shedding for each Reynolds number is different. By taking a Fourier transform of the time evolution of a data point located at (0.5,8), we find that for the original datasets the vortex shedding frequency is 0.448 Hz, 1.260 Hz, 1.380 Hz, 1.388, and 1.401 Hz for Re=100, 300, 600, 800, and 1000 respectively, and for the E-GP models the frequencies are 0.452 Hz, 1.21 Hz, 1.36 Hz, 1.36 Hz, and 1.36 Hz respectively.

Generalizing from Learned Dynamics to Unknown Dynamics

Having seen that it is possible to find a single transition in the weight space that models the dynamics systems over a range of parameters, the next challenge is to be able to model flows with parameters that the model has not been trained on. We derived an $\widehat{A}$ matrix from the Re=100, 300, 600, and 1000 datasets and tested it against the Re=800 dataset. The results are below in Figure S8(b). For the first 120 snapshots, the total percentage error remains under 10%, which is satisfactory. After this, however, the total percentage error curves upwards as the slight errors in the transition matrix compound. Over 800 snapshots, we found an average total percentage error of less than 25%.

Linear Dynamical Layer Analysis & Insights

Due to the spatial encoding of the weights which the linear transition model operates on, we are able to analyze the dynamics and find physical insights into the process. We demonstrate two techniques: (1) using eigendecomosition of the transition matrix to discover the eigenfunctions

and invariant subspaces of system, and (2) visualizing the most significant spatial interactions in the system.

By marking which kernel centers are associated with different invariant subspaces, we can spatially separate the space into multiple dynamic modules. The physical insight is that some areas of the space are dynamically entangled with each other, and other are independent. For those interested in monitoring spatiotemporally evolving systems, the number and location of the invariant subspaces determines how many and where feedback sensors ought to be for robust prediction of the system state.

Before attempting a Jordan decomposition of $\widehat{A}$, we zero any elements smaller than some small ε in order to stabilize the algorithm for matrices with many elements close to zero. Afterwards we visualize the eigenvector matrix using a *logarithmic* color chart, as seen in Figures S9(a),S9(b),S9(c). These plots are for models trained individually on Re=100 and Re=1000 with 300 kernels, and on all five with 600 kernels, for comparison. We see three categories of eigenvector in the rows: (1) Rows at the bottom that have exactly one non-zero elements, (2) In the middle, a couple rows with a dozen significant elements, and (3) at the top a number of rows that affect the majority of the kernel centers in the space.

Each eigenvector of (1) spans its own invariant subspace, and is depicted in magenta circles in Figure S10. Category (3) is one invariant subspace, depicted with black crosses. Category (2) is subsumed in Category (3). The figures show that the dynamics near/around the cylinder and in its wake are so entangled that a single sensor measurement in that area may be sufficient to estimate over that entire subspace. On the other hand, areas far from the core of dynamic excitement are their own independent, invariant subspaces, and thus must be monitored locally.

Another way to visualize the operation of the linear transition matrix is to plot lines between kernel centers that are influencing each other strongly. That is, if we draw a line center c_j to c_i for each of the (relatively) largest elements a_{ij} of $\widehat{A}$, one can see how the system dynamics are coupled spatially (Figures S11(a),S11(b),S11(c)). We can also plot the magnitude of a_{ij} in a third axis for further insight into the most dominant dynamic connection in the system.

Author Biography

Joshua E. Whitman is a PhD student at the University of Illinois in Urbana-Champaign. He received Bachelors of Science degrees in Mechanical Engineering and Mathematics from Oklahoma State University in 2015, and his Masters of Science in Mechanical Engineering from the University of Illinois in 2018. His research is centered in the fields of machine learning, controls, and autonomy, and his work focuses on developing new methods for learning, monitoring, and controlling the dynamics of complex spatiotemporally-evolving systems.

Harshal Maske completed his PhD in 2018 from the University of Illinois in Urbana-Champaign, he is currently a Research Engineer with Ford Motor Company. Prior to joining Ford, Harshal was a research intern at Mitsubishi Electric Research Laboratory. He had received his Masters and Bachelors integrated degree in Mechanical Engineering from the Indian Institute of Technology Kharagpur, India, in 2009. After completing his masters, he worked for three years at Defense R&D Organization (2009-2012), India and for one year at John Deere (2012-2013).

Hassan A. Kingravi is a senior data scientist at MailChimp, where he conducts research on machine learning and builds systems for fighting fraud. His research interests revolve around the interplay of control theory, signal processing, and machine learning for spatiotemporal data. He received his PhD in Electrical and Computer Engineering from the Georgia Institute of Technology in 2014 and did a postdoc at Oklahoma State University with Chowdhary's group.

Girish Chowdhary is an assistant professor and Donald Biggar Willet Faculty fellow at the University of Illinois at Urbana-Champaign and affiliated with Electrical and Computer Engineering, Agricultural and Biological Engineering, Computer Science, Aerospace Engineering, and is a member of the UIUC Coordinated Science Laboratory (CSL). He is the director of the Distributed Autonomous Systems laboratory and the Field Robotics Engineering and Science Hub (FRESH) at UIUC. He holds a PhD (2010) from Georgia Institute of Technology in Aerospace Engineering. He was a postdoc at the Laboratory for Information and Decision Systems (LIDS) of the Massachusetts Institute of Technology for about two years (2011-2013). He was an assistant professor at Oklahoma State University's Mechanical and Aerospace Engineering department (2013-2016). Prior to joining Georgia Tech, he also worked with the German Aerospace Center's (DLR's) Institute of Flight Systems for around three years (2003-2006). His undergraduate institution was the Royal Melbourne Institute of Technology in Australia. Girish's ongoing research interest is in theoretical insights and practical algorithms for adaptive autonomy with applications in field robotics.

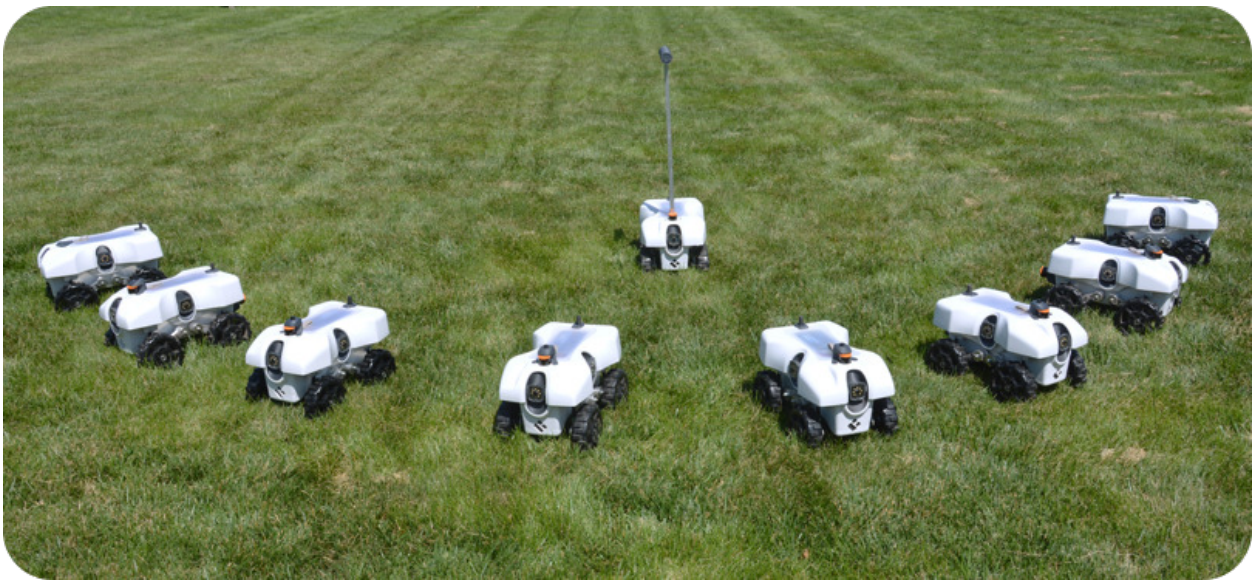


Figure S1: The TerraSentia robots developed by Chowdhary's group at UIUC and commercialized by EarthSense inc. Agricultural robots such as these present exciting possibilities for distributed agricultural management with teams of compact, ultra-light, under-canopy robots equipped with advanced autonomy and machine learning. Such robots can fill the niche between large farm equipment and manual labor, as well as enable perennial polycultures, and closer to home, they could also weed your garden! Advances in persistent multi-agent autonomy in harsh, changing, and uncertain environments driven by the controls community are driving such exciting possibilities of the future of agriculture.

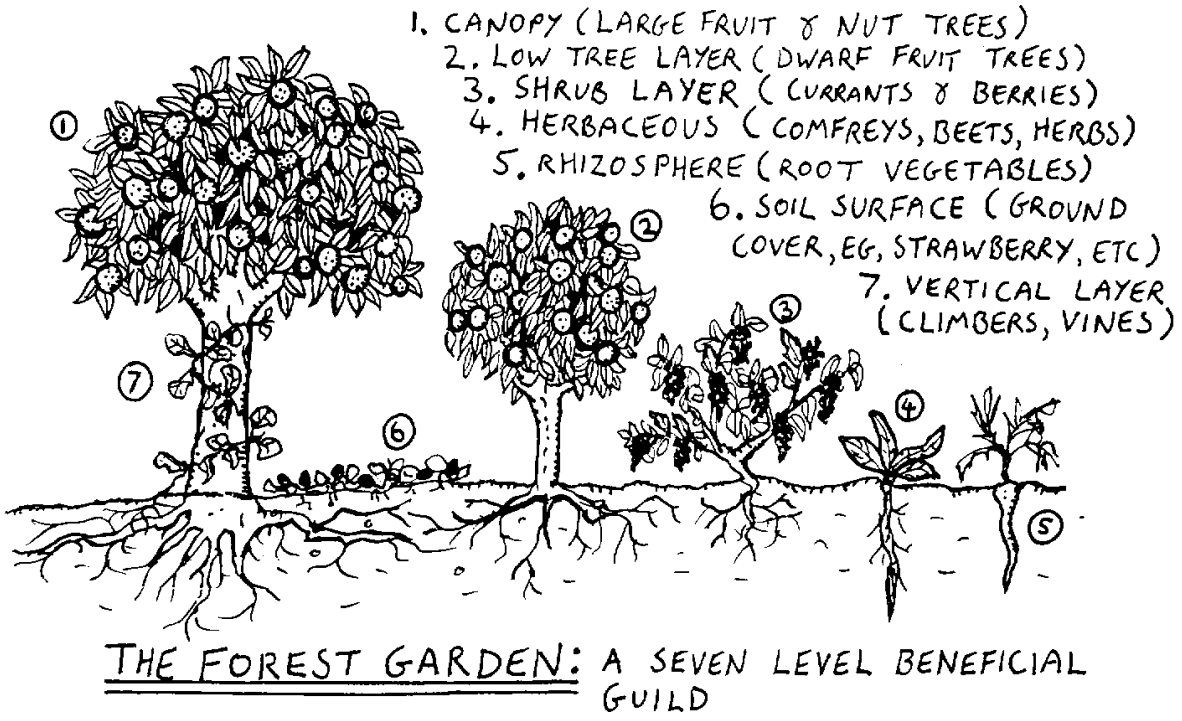
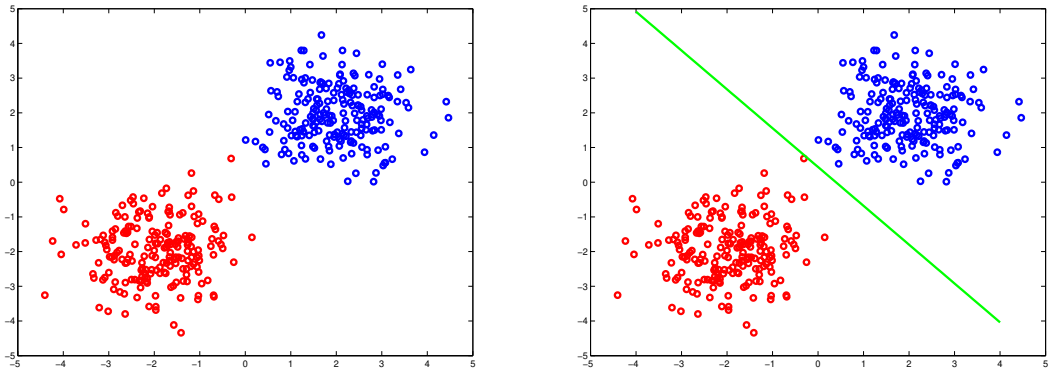


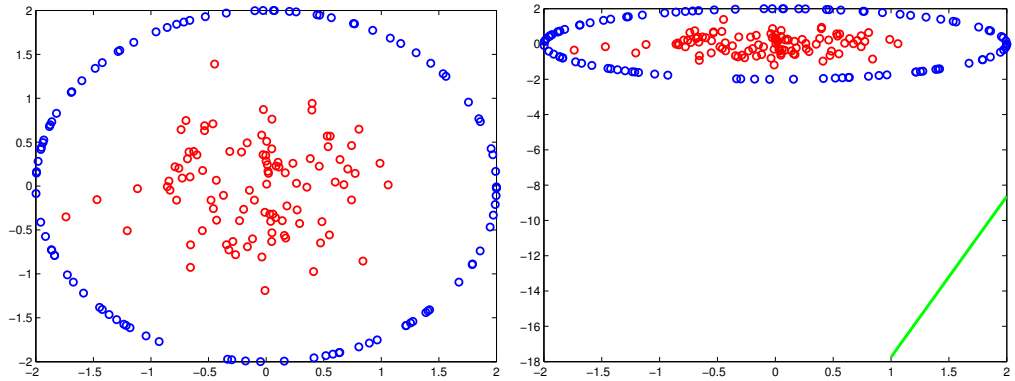
Figure S2: The seven layers of the forest garden. Polyculture agricultural production systems are designed ecosystems that can be more productive and sustainable than traditional monoculture systems. Managing such complex systems requires fundamental advances in robotics, spatiotemporal modeling, and control of complex biological processes, all areas where the control community can help. Figure source [97] see also [98].



(a) Data from classes A and B.

(b) Linear boundary separating data.

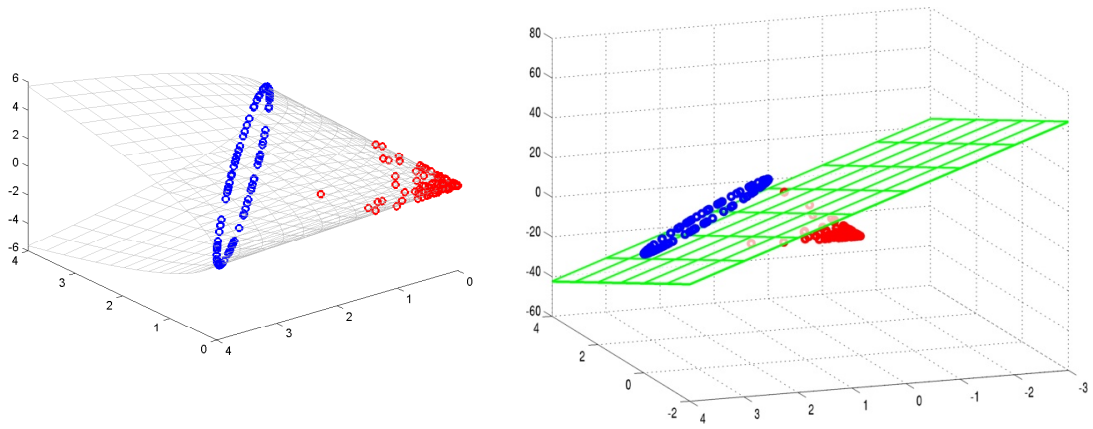
Figure S3: Example of linearly separable data. Any simple linear learning algorithm e.g. perceptron, finds a solution.



(a) Data from classes A and B.

(b) Linear boundary fails to separate data.

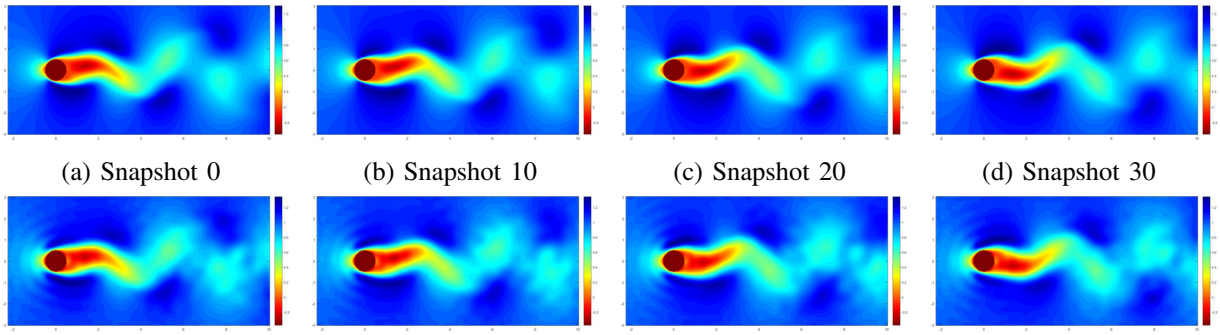
Figure S4: Example of nonlinearly separable data. Perceptron fails to find a solution, and diverges.



(a) Nonlinear mapping of data.

(b) Linear boundary in new space.

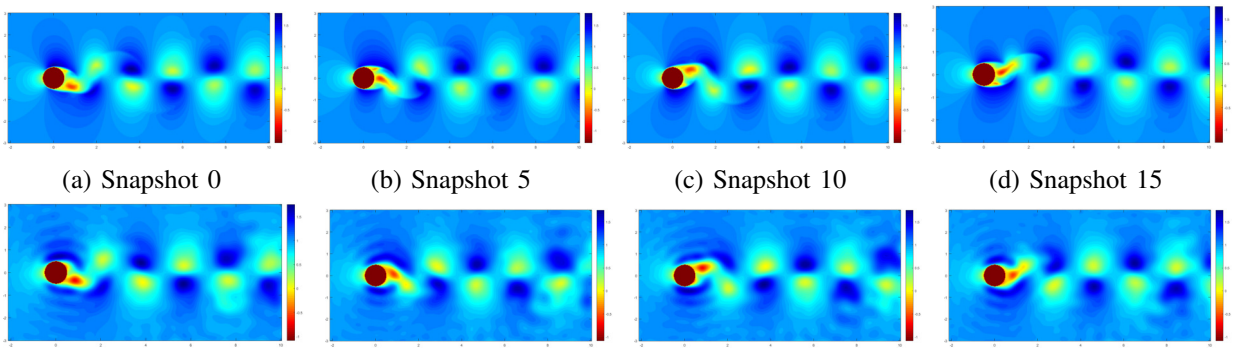
Figure S5: If we map the same data using a nonlinear map $\phi(x, y) := (x^2, y^2, 2xy)$, the perceptron now finds a solution in 3 dimensions.



(a) Snapshot 0 (b) Snapshot 10 (c) Snapshot 20 (d) Snapshot 30

(e) Snapshot 0 (f) Snapshot 10 (g) Snapshot 20 (h) Snapshot 30

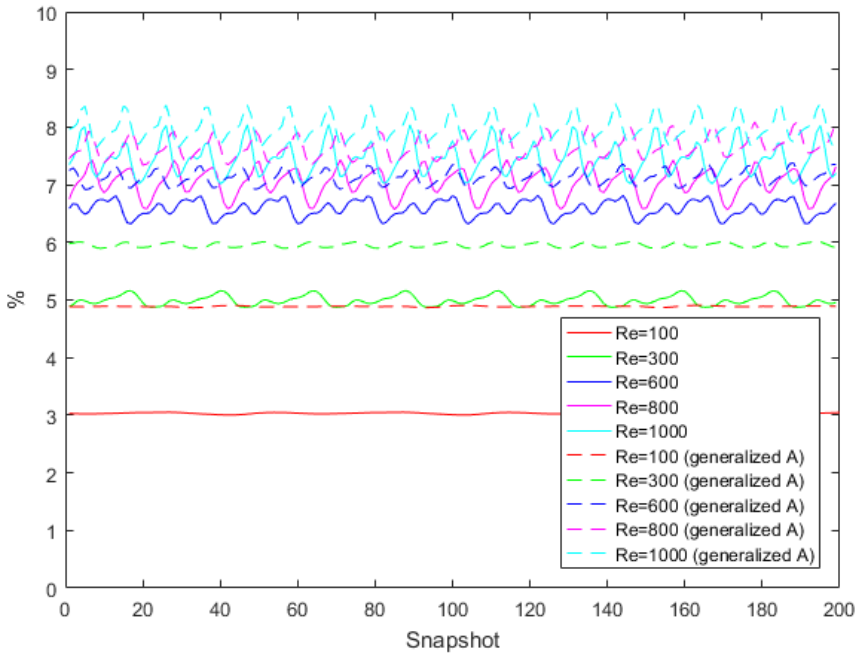
Figure S6: Visualization of Fluid Flow at Re = 100, CFD (a-d), E-GP (e-h)



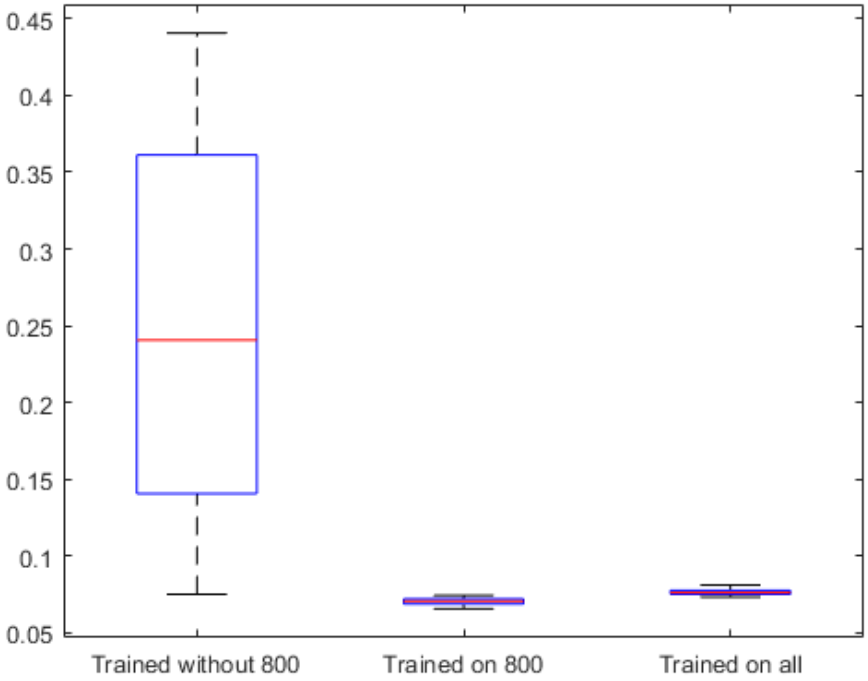
(a) Snapshot 0 (b) Snapshot 5 (c) Snapshot 10 (d) Snapshot 15

(e) Snapshot 0 (f) Snapshot 5 (g) Snapshot 10 (h) Snapshot 15

Figure S7: Visualization of Fluid Flow at Re = 1000, CFD (a-d), E-GP (e-h)

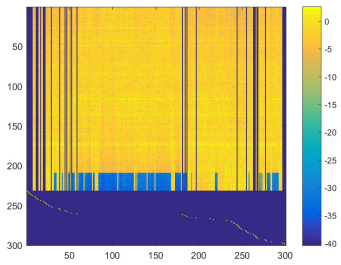


(a) Universal Generalizer vs Individual Models

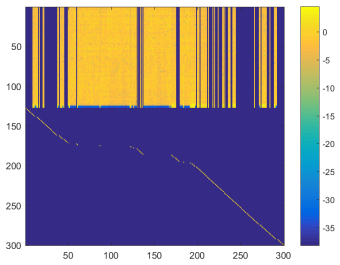


(b) Different Models Tested on Re=800

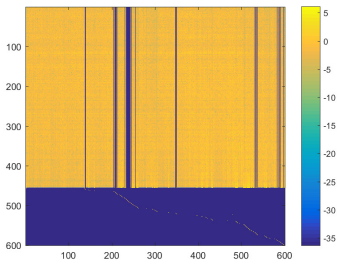
Figure S8: Total Percentage Errors



(a) Re = 100, ε = 0.005

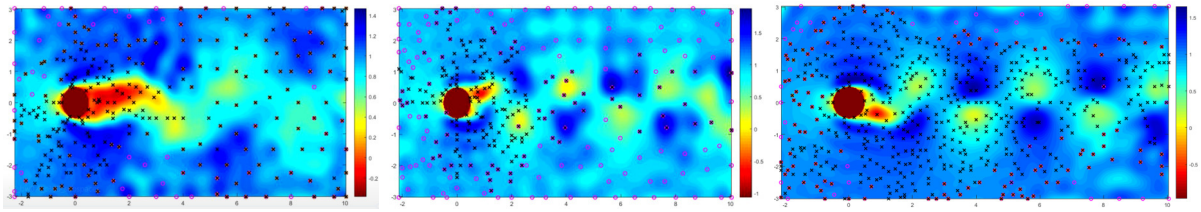


(b) Re = 1000, ε = 0.05



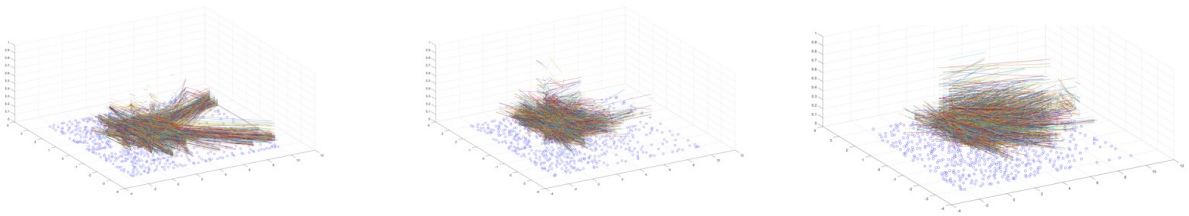
(c) All Reynolds numbers, ε =0.069

Figure S9: Eigenvector Heat Maps



(a) Re = 100, ε = 0.005 (b) Re = 1000, ε = 0.05 (c) All Reynolds numbers, ε = 0.069

Figure S10: Invariant Subspaces



(a) Re=100 (b) Re=1000 (c) Trained on all 5 datasets

Figure S11: Visualization of Co-Relations in Transition Matrix