Design of Digital Twin Sensing Strategies via Predictive Modeling and Interpretable Machine Learning

Michael G. Kapteyn[*]
University of Texas at Austin, Austin, TX 78712

Karen E. Willcox [†]
University of Texas at Austin, Austin, TX 78712

This work develops a methodology for sensor placement and dynamic sensor scheduling decisions for digital twins. The digital twin data assimilation is posed as a classification problem, and predictive models are used to train optimal classification trees that represent the map from observed data to estimated digital twin states. In addition to providing a rapid digital twin updating capability, the resulting classification trees yield an interpretable mathematical representation that can be queried to inform sensor placement and sensor scheduling decisions. The proposed approach is demonstrated for a structural digital twin of a 12ft wingspan unmanned aerial vehicle. Offline, training data are generated by simulating scenarios using predictive reduced-order models of the vehicle in a range of structural states. These training data can be further augmented using experimental or other historical data. In operation, the trained classifier is applied to observational data from the physical vehicle, enabling rapid adaptation of the digital twin in response to changes in structural health. Within this context, we study the performance of the optimal tree classifiers and demonstrate how they enable explainable structural assessments from sparse sensor measurements, and also inform optimal sensor placement.

I. Introduction

Digital twins have emerged as a paradigm for characterizing, monitoring, analyzing, and optimizing complex systems by combining state-of-the-art computational models with data-driven learning [1–3]. A digital twin is a computational model (or a set of coupled computational models) that evolves over time to persistently represent the structure, behavior, and context of a unique physical asset such as a component, system, or process [4]. Digital twins have clear potential to underpin the intelligent automation of the future by enabling asset-specific data-driven decision-making. For engineering systems, incorporating physics-based modeling within the digital twin is critical to providing reliable predictive capability, which in turn drives decision-making. However, in order to achieve the level of specificity promised by the digital twin paradigm, such models must be regularly adapted so as to ensure that the digital twin represents the unique characteristics of a physical asset at a specific point in its lifecycle. Thus, the design of digital twin sensing strategies plays a critical role. The contribution of this paper is the development of a methodology for designing explainable digital twin sensing and model updating strategies by leveraging machine learning classification methods coupled with physics-based models of the system.

The promise of digital twins has been recognized across a wide range of application areas. Engineering applications include virtualized structural health monitoring, simulation-based certification, and predictive maintenance [5–8], as well as optimized fleet management [6, 9, 10]. Beyond engineering, healthcare digital twins promise to advance medical assessment, diagnosis, personalized treatment, and *in-silico* drug testing [11–14], educational digital twins offer a path to personalized education [15], and smart cities enabled by digital twins promise to revolutionize urban planning, resource allocation, sustainability, and traffic optimization [16].

The proliferation of sensors and increasing connectivity among physical assets means that engineering systems now produce large amounts of readily accessible asset-specific data. For example, modern aircraft have been predicted to produce approximately 1 petabyte of data each flight-hour by 2025 [5]. The ability to assimilate these data in order to rapidly extract valuable insights and incorporate them into the digital twin is necessary to avoid the so-called

[*] Postdoctoral Fellow, Oden Institute for Computational Engineering and Sciences

[†] Director, Oden Institute for Computational Engineering and Sciences, Professor of Aerospace Engineering and Engineering Mechanics

curse of history, in which the amount of data requiring storage grows *ad infinitum* as time goes on. Performing this assimilation in a rapid, yet principled, manner while accounting for uncertainty is crucial to maintaining the accuracy and reliability of the digital twin. Prior works on data-driven digital twin updating have typically focused on fully Bayesian methods [7, 17, 18] or optimization-based machine learning methods, targeting either surrogate model fitting [19], model calibration [20], or learning a model correction term [21–23].

Ensuring that the digital twin is reliable demands that any data-driven methods influencing the digital twin be both accurate and interpretable. Interpretability ensures that the digital twin can be validated and/or certified, and offers insight into which data are most informative, and how exactly those data are being used in the digital twin. While Bayesian methods and direct model calibration methods typically involve updating a set of interpretable model parameters, these methods require a number of computational model evaluations which is typically large, and which scales with the dimension of the model parameter space. This makes such methods computationally intractable in many applications due to a combination of performance requirements (e.g., when data must be assimilated rapidly to enable critical decision-making) and resource constraints (e.g., limited computing or electrical power when a digital twin exists on an onboard computer or edge device). On the other hand, machine learning surrogate modeling approaches can be computationally efficient, but are typically treated as black-boxes and can thus be difficult to understand, validate, and certify.

In a previous work, we introduced the idea of using decision tree classifiers to update a digital twin based on incoming sensor data [24]. This paper serves to further develop this idea into a complete methodology for the design of digital twin sensing strategies. In particular, we here consider various additional aspects of the sensing task for predictive digital twins, namely, how decision trees can be used for sensor placement, dynamic sensor selection, and sensor sensitivity analysis. The proposed methodology combines physics-based modeling with machine learning. The physics-based models that comprise the digital twin are used to simulate asset operation during an offline phase, generating a training dataset comprising expected sensor measurements for various asset states. We then use this dataset to train an interpretable classifier that takes as input online sensor data and outputs an estimate of the underlying asset state by selecting the physics-based model configuration that best explains the observed sensor data. Deploying such a classifier within the digital twin enables rapid data-driven updating of the computational models comprising the digital twin.

The machine learning approach explored in this work is based on optimal decision trees [25, 26]. In particular we show how optimal decision trees can be trained that leverage only a subset of the input observations. This capability enables the methodology to reveal which observations are most useful for a given classification task. As training data can be simulated prior to sensor installation and asset deployment, this enables us to leverage optimal trees for sparse sensor placement and/or dynamic sensor scheduling in support of an accurate and efficient digital twin. Whereas existing sensor placement approaches [27, 28] typically focus on a single output quantity of interest, our approach can naturally produce classifiers that act on multi-modal measurement data. In addition to achieving state-of-the-art classification accuracy, a key feature of optimal decision trees approach is interpretability. In particular, since the approach does not require data compression or reduction of the observation space, it produces a classifier that maps directly from the observation space to the digital twin state space, while also revealing an explicit partitioning of the space of observational data. This makes sensor placement, as well as digital twin state updates, interpretable since is clear which sensors are contributing to a given classification decision and precisely where applied decision boundaries lie. This interpretability can facilitate verification and validation of a digital twin, as well as promoting confidence in decisions or analysis provided by the digital twin.

This paper is organized as follows. Section II establishes the mathematical abstraction of a digital twin, as well as associated definitions and notation, that we adopt in this work. Section III formulates the proposed approach to data-driven model adaptation via optimal classification trees. Section IV applies the proposed methodology to an example application, namely the sensing strategies and associated model adaptation of a structural digital twin for an Unmanned Aerial Vehicle (UAV). Finally, Section V summarizes the contributions of this work and highlights areas requiring future study.

II. Mathematical abstraction of a digital twin

This section serves to place the sensor design methodology proposed in this work into the broader context of a complete digital twin system. To do this, we adopt the mathematical abstraction for digital twins proposed in [29]. In this abstraction, we define six quantities as shown in Table 1. These quantities describe the coupled system comprised of the physical asset and its associated digital twin. In a typical digital twin application, each of these six quantities will be considered to vary with time, that is, the physical state s, digital twin state d, quantities of interest q, observational data o, control inputs u, and rewards r are all modelled as time-dependent variables. For ease of exposition, and to simplify later notation, we will not express this time dependence explicitly in the discussions below.

In this section, we further define each element of our mathematical representation of a digital twin, and explain how each element factors into the design of digital twin sensing strategies.

Table 1 Summary of the six quantities comprising our abstraction of an asset-twin system.

Quantity	Notation	Description
Physical State	$s \in \mathcal{S}$	Parametrized state of the physical asset
Digital Twin State	$d \in \mathcal{D}$	Configuration of computational models comprising the digital twin
Quantities of Interest	$q \in \mathcal{Q}$	Quantities describing the asset, estimated via model outputs
Observational Data	$o \in \mathcal{O}$	Available information describing the state of the physical asset
Control Inputs	$u \in \mathcal{U}$	Actions or decisions that influence the physical asset
Reward	$r \in \mathcal{R}$	Quantifies overall performance of the asset-twin system

Physical state and digital twin state. The physical twin refers to the physical component, system, or process for which we are creating a digital twin in support of monitoring and decision-making. In general, the physical state, $s \in \mathcal{S}$, will be only partially and indirectly observable through observational data. Here $\mathcal{S}$ denotes the state space of the physical twin. We consider systems whose state evolves over time within this state space. The digital twin is a parametrized model or set of coupled models that evolve over time to persistently represent the physical twin. The digital twin state, $d \in \mathcal{D}$, comprises the parameters of these models. Note that d includes typical state variables, such as position and velocity of a vehicle, but also other parameters (e.g., geometry, structural health, fuel state) that define the current state of the system, as represented in the digital twin models. Here $\mathcal{D}$ denotes the digital twin state space, i.e., the space of all possible values for the digital twin state. When designing digital twin sensing strategies, we must ensure that the digital twin state can be inferred with sufficient accuracy and robustness. There is thus a rich two-way interaction between the selection of the digital twin state and the observational data: the digital twin state space must be defined with sensing constraints in mind, while observability of the digital twin state will factor into sensor design and scheduling decisions.

Observational data. The digital twin state is informed by observational data received from the physical twin. The observational data, $o \in \mathcal{O}$, might include sensor data, physical inspection data, information from diagnostic or error reporting systems, etc. Here $\mathcal{O}$ denotes the space of possible observations. Digital twin sensing strategies influence observational data through sensor placement and design (i.e., which sensors are placed onboard the system and where) and sensor selection (i.e., which sensor data are processed in any given situation, including dynamic sensor scheduling). We note that in some cases, the sensing design problem will lead to a formulation where the space $\mathcal{O}$ itself changes in time (e.g., in the case where we make the decision to add a sensor).

Control. The control input, $u \in \mathcal{U}$, where $\mathcal{U}$ denotes the space of possible control inputs, defines actions that can influence the physical state in some way. This could involve an action or intervention that changes the instantaneous physical state or changes the operating conditions of the asset, thereby influencing the state evolution dynamics. Similarly, control inputs could also influence the observational data that is generated by an asset, either instantaneously (e.g., deciding to perform an inspection of the asset) or in the future (e.g., installing a new sensor or otherwise modifying the sensor architecture). It is this latter class of control inputs (which we refer to as different digital twin sensing strategies) that we focus on in this work.

Quantities of interest. If the digital twin state is thought of as an encapsulation of important inputs to the computational models comprising the digital twin, then quantities of interest, $q \in \mathcal{Q}$ are an encapsulation of the outputs of these models. The quantities of interest thus depend directly on the digital twin state, but can only be evaluated via the computational models comprising the digital twin. Here $\mathcal{Q}$ denotes the space of possible values that the quantities of interest may take. Quantities of interest help to characterize the physical asset by serving as computational estimates of quantities that may not be directly observable, for example, the full stress or strain field within a structure. Quantities of interest drive sensor placement and scheduling decisions by defining the key prediction targets that will ultimately inform decision-making. Optimal sensing decisions must factor in these targets in a goal-oriented way.

Reward. The reward, $r \in \mathcal{R}$, quantifies the value of different states of the complete asset-twin system. Here $\mathcal{R}$ denotes the space of possible reward values. The reward model could encode various success or failure states for the system, various costs associated with acquiring observational data or applying control inputs, or a measure of accuracy and reliability of the digital twin. Note that the reward might represent a real-world quantity such as cost, or it could be a more abstract quantity defined and tuned to achieve the desired system performance. Similar to its use in agent-based models such as partially observable Markov decision processes (POMDPs), the reward can serve as the quantity that we seek to optimize when deciding on a control strategy using the digital twin. However, note that in some cases digital twins may be used for asset monitoring only (by applying a fixed predefined control policy). Moreover, in the digital twin context the choice of reward may vary over time, for example, the reward defined during design, manufacturing or calibration may differ from the reward defined for an operational phase.

This mathematical abstraction provides the framework on which to pose the task of designing digital twin sensing strategies, namely, the tasks of digital twin sensor placement and sensor scheduling. Quantifying the reward associated with sensing decisions must be done in an integrated way that factors in the role of observational data in updating the digital twin state, as well as the cost associated with a given sensing decision. Our past work addressed this by using a dynamic decision network to model the relationships between the six elements of our digital twin mathematical abstraction [29]. Figure 1 shows an example of the structure of such a network where observational data generated by the physical twin is used to infer the digital twin state, which in turn is used to estimate quantities of interest, drive decisions and compute rewards.

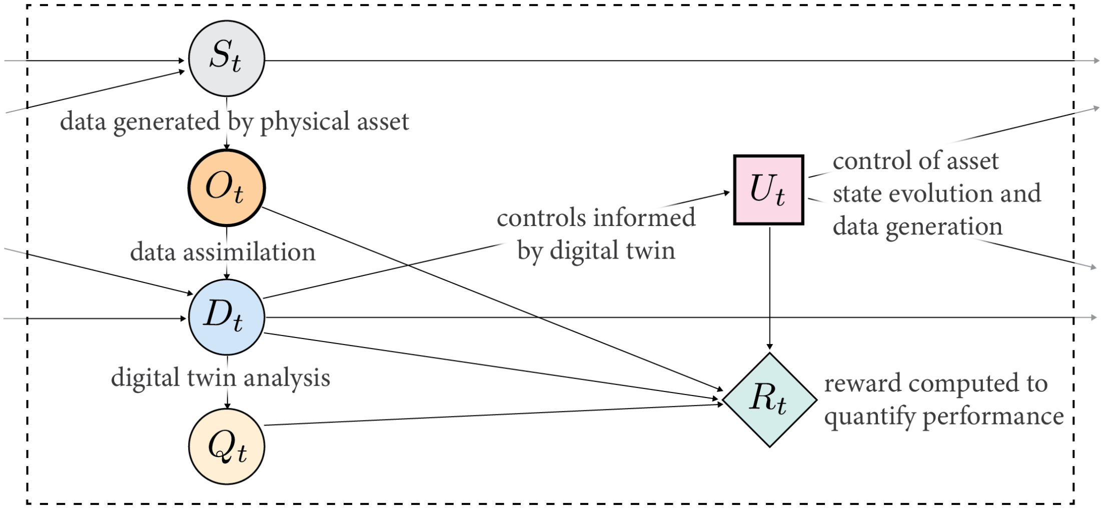


Fig. 1 Example structure of a network used to model the relationships among the elements of the digital twin abstraction. In this example, each quantity is modeled as a random variable (denoted with capital letters), the subscript t denotes quantities at a time stage t, and the relationships are modeled via a dynamic decision network.

The probabilistic graphical model of [29] is one approach to integrated modeling of the digital twin data, models, and decisions. Using this formalism, the digital twin updating is cast as a Bayesian state estimation task [30]. Another approach, which we employ here, is to consider the same information flow as in Figure 1, but instead cast the data assimilation task as a classification problem. While the classification formulation does not represent uncertainty as integrally as the probabilistic graphical modeling formulation, it has other advantages in being more interpretable, traceable, and providing a rapid online evaluation capability. To this end, the next section develops a formulation that uses optimal classification trees as the mathematical and computational foundation on which to address the data assimilation, sensor placement, and sensor scheduling tasks.

III. Digital twin adaptation and sensing using optimal classification trees

This section presents a methodology for digital twin data assimilation, sensor placement and sensor scheduling decisions. We first motivate the use of optimal classification trees (OCTs), then describe methodology for classification via decision trees and a scalable approach for learning OCTs. We then present the formulation of digital twin sensing strategy design using OCTs.

A. Problem framing and motivation

In the context of the digital twin mathematical abstraction of Section II, our goal is to design sensing strategies that optimally influence observational data o so that we can successfully infer the digital state d, achieve sufficiently accurate estimates of the quantities of interest q, and maximize rewards r. Here, sensing strategies are defined by our control actions, u, which encompass both sensor placement decisions, which would likely take place in an upstream design task but could also reflect an in-service modification, and sensor scheduling decisions, which take place dynamically during operation. Figure 2 depicts each of these tasks in the digital twin context.

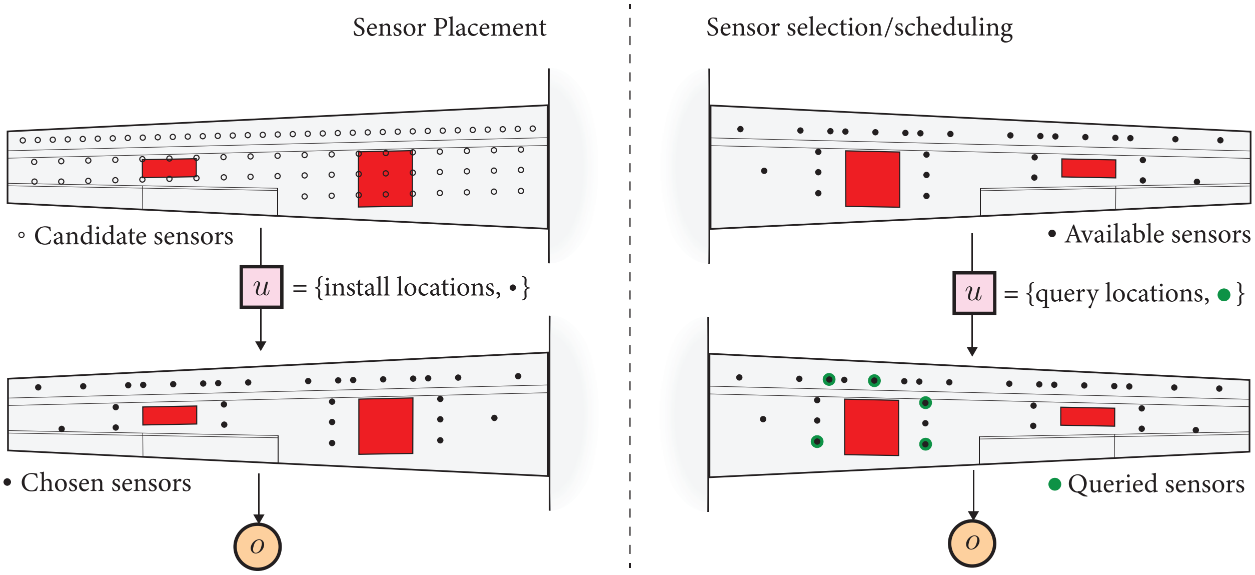


Fig. 2 Graphical depiction of different sensing tasks, each of which constitute an aspect of designing a digital twin sensing strategy. The depicted application is revisited in Section IV.

In this work, we employ OCTs as the mathematical and computational foundation for our methodology. We make this choice for several reasons. First, formulating the digital twin updating as a classification problem leads to an offline/online decomposition of tasks that enables rapid online evaluation at timescales useful for practical deployment in operation. Second, the offline training of the OCT can be achieved for high-dimensional digital twin state spaces using a recently developed mixed-integer optimization formulation that leads to a scalable training algorithm [25, 26]. Third, the OCT formulation leads to a methodology that is traceable, in the sense that one can easily diagnose which sensors have contributed to informing the digital twin, and what decision boundaries were applied to their readings, allowing a human to interpret the classification result in the context of the physical system. As a result, this method lends itself well to decision processes that ultimately involve a mixture of human-driven and automated decisions.

B. Classification via decision trees

Decision tree methods are widely used in machine learning and statistics [31, 32] as a way to learn from a set of training data a recursive, hierarchical partitioning of the feature space (the space of observational data O in our setting) into a set of disjoint subspaces. We here consider *classification trees*, in which each region is assigned a single deterministic label assignment (typically the most common label among training data assigned to the region). To illustrate this idea, we use a classical demonstrative dataset for classification problems: Fisher's iris dataset [33] which is available from the UCI machine learning repository [34]. In this example, the training data consists of $N_t = 150$ observation-state pairs. For this dataset there are three possible digital states, $\mathcal{D} := \{d_1, d_2, d_3\}$, with $s = 50$ observations for each state, o_j^k, $j = 1, \ldots, 3$, $k = 1, \ldots, 50$. Here each observation is a four-dimensional vector of real numbers, i.e., $O := \mathbb{R}_s^N$ with $N_s = 4$ observed quantities. We refer to the observed quantities as Features A, B, C, and D. Recall that in our context, each of these features would be a component of the observational data, for example a measurement from a single sensor or a particular result from an inspection.

In Figure 3, we plot the dataset according to the value of features A and B. We then show examples of using either axis-aligned or hyperplane splits to partition the feature space. We also show the classification trees that correspond to each partitioning.

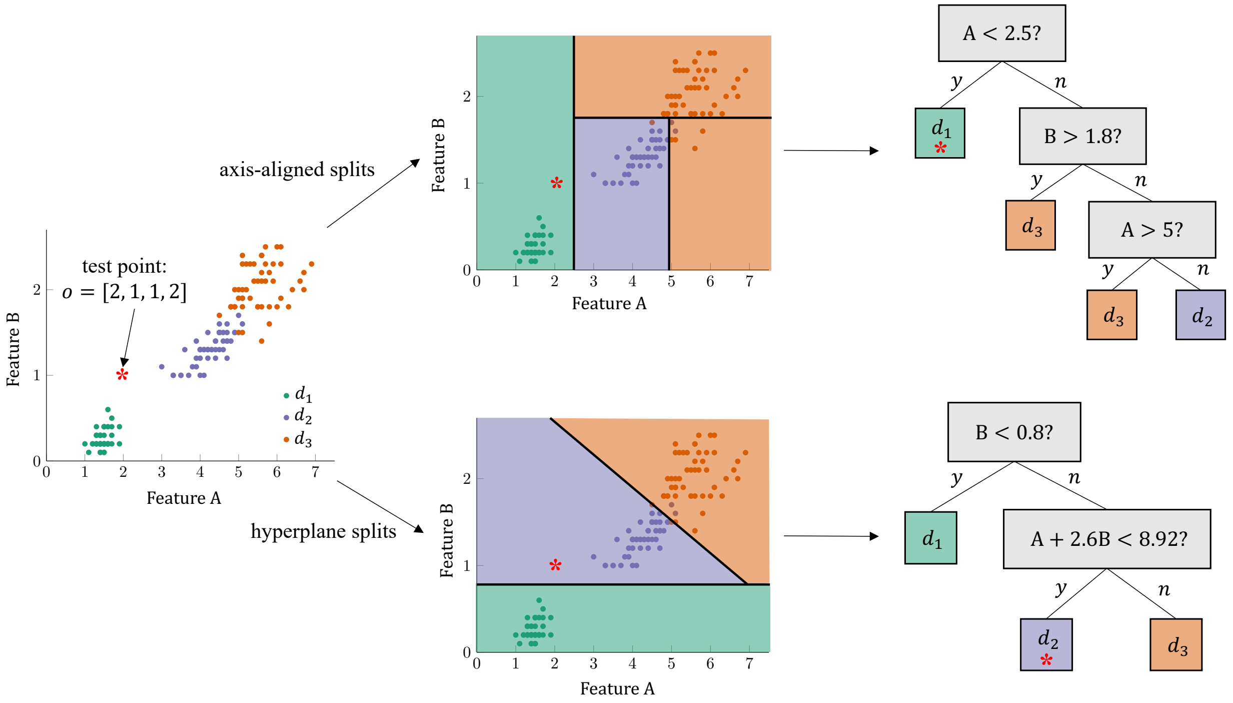


Fig. 3 Two possible ways to recursively partition an illustrative dataset, one using axis-aligned splits only, and another allowing for more general hyperplane splits. The corresponding classification tree is shown for each partitioning. A test point is shown to illustrate how a new datapoint is assigned a label using the classification tree.

Once a classification tree, T, has been generated, it can be used to predict outcomes for new, previously unseen observations. When a test observation, o, is obtained, evaluation begins at the root node of the tree. At each branching node in the tree, a decision is made depending on the observed feature values, until the test observation reaches a leaf node, at which point a predicted output is assigned. For example, given the test observation $o = [2, 1, 1, 2]$, depicted as an asterisk in Figure 3, $T(o) = d_1$ using the classification tree with axis-aligned splits, while $T(o) = d_2$ if we instead use the classification tree with hyperplane splits. Recall that in our context these labels would be interpreted as a specific digital state that, when used as input to the physics-based models comprising the digital twin, best explains the observation o.

Hyperplane splits involve a linear combination of multiple features, and thus can be less interpretable than axis-aligned splits. However, since they are more expressive than axis-aligned splits, trees with hyperplane splits can be shallower for a given level of accuracy. Therefore, incorporating hyperplane splits might in fact improve interpretability of classification tree for a given accuracy; a shallow tree with hyperplane splits may be more interpretable than a deep tree with axis-aligned splits, especially if the features used in the hyperplanes are somehow related.

Recall that in this work we limit our focus to classification trees in which each leaf is assigned a single predicted output. This results in a piecewise constant function T and is thus amenable to discrete output spaces (in this case discrete digital state spaces). However, we note that our approach can be generalized to predict continuous outputs by instead using *regression trees*. In regression trees, each leaf is fitted with a continuous prediction model, e.g., a linear regression model or polynomial regression model.[35] When a test observation reaches a leaf node, the appropriate regression model is applied to the observation to determine the final prediction. While typically being more computationally expensive to train, regression tree models can thus result in more expressive decision models, e.g., T can be piecewise polynomial over the feature space O.

C. Optimal classification trees

The question remains of how to generate a classification tree for a given task. In this work we adopt a scalable method for learning an *optimal* classification tree from data, using the mixed-integer optimization formulation of the problem developed by Bertsimas and Dunn [25, 26]. The optimization decision variables include all the quantities necessary to define the decision tree, which we denote by T. This includes the coefficients defining the hyperplane splits for each decision node, as well as variables that determine the arrangement of nodes in the tree. The objective of the optimization problem is to construct a tree that minimizes the misclassification error on the training set, while also minimizing the complexity of the tree in order to improve interpretability and avoid overfitting, thereby promoting good out-of-sample performance. This objective function can be stated as:

$$\min_T R(T) + \kappa|T| \tag{1}$$

where T is the classification tree, $R(T)$ is the misclassification error of the tree on the training data, and $|T|$ is the complexity of the tree, as measured by the number of splits. The complexity parameter κ governs the tradeoff between accuracy and complexity of the tree. The full problem, including the definition of variables comprising the tree, T, and the constraints required to enforce the tree structure can be found in Ref. 25. This optimization problem can be solved directly using commercial solvers such as Gurobi [36]. However, the direct solution approach generally scales inefficiently, as the number of integer decision variables increases exponentially as the depth of the tree increases, and linearly as the number of training data points increases. To overcome this, an efficient local-search method has been developed which is able to efficiently solve the problem to near global optimality for practical problem sizes, in times comparable to previous methods [26]. In particular, under a set of realistic assumptions, the cost of approximately solving (1) with parallel splits using the local-search method is $O(|\mathcal{D}|\, N_s\, N_t\, \log\, N_t)$[37], which is only a factor of $\log\, N_t$ greater than the classical method known as Classification and Regression Trees (CART) [38], which generates classification trees using a sub-optimal greedy approach.

In addition to computational efficiency, OCTs to achieve state-of-the-art accuracy comparable to ensemble tree methods such as Random Forests [39] or gradient boosted trees [40] while maintaining the interpretability that comes from using only a single decision tree. In addition to being near globally optimal in-sample, it is shown in Ref. 37 that OCT-H outperforms CART and Random Forests in terms of out-of-sample accuracy, across a range of 60 practical classification problems from the University of California, Irvine (UCI) machine learning repository [34] and at all practical values for the tree depth. As a final comparison to other state-of-the-art methods, we note that Ref. 26 compares optimal trees with various neural network architectures. In particular, they show that various types of feedforward, convolutional, and recurrent neural networks can be reformulated exactly as classification trees with hyperplane splits. This result suggests that the modeling power of an OCT with hyperplane splits is at least as strong as these types of neural network, while being easily interpretable.

D. Design of digital twin sensing strategies via predictive modeling and optimal classification trees

Figure 4 presents an overview of the proposed methodology which combines the physics-based models comprising a digital twin with optimal classification trees to discover computationally efficient digital twin sensing strategies.

Recall that in the digital twin setting, the OCT represents a mapping from the space of observational data to the digital state space, i.e.,

$$T : \mathcal{O} \to \mathcal{D}. \tag{2}$$

The OCT, T, is learned from data by leveraging the digital twin models during an offline phase as described below. Then, during the online phase, this mapping is used to perform data assimilation by applying it to newly acquired observational data, such that we make the estimate

$$d = T(o) \in \mathcal{D}. \tag{3}$$

Digital twins rely on predictive models as the basis for their predictive capability. For engineering systems, such predictive models are typically physics-based. As discussed in Section I, the use of predictive physics-based models for the solution of inverse problems in this way is commonplace, however using these models directly typically requires many model evaluations which is computationally intractable in an online setting. Instead, we leverage these physics-based predictive models offline in order to generate a training dataset of digital twin state and observation pairs, to which we can apply the OCT methodology described in the previous subsection. Note that this training dataset represents the physical relationship between digital twin states and their corresponding sensor observations, as encoded in the

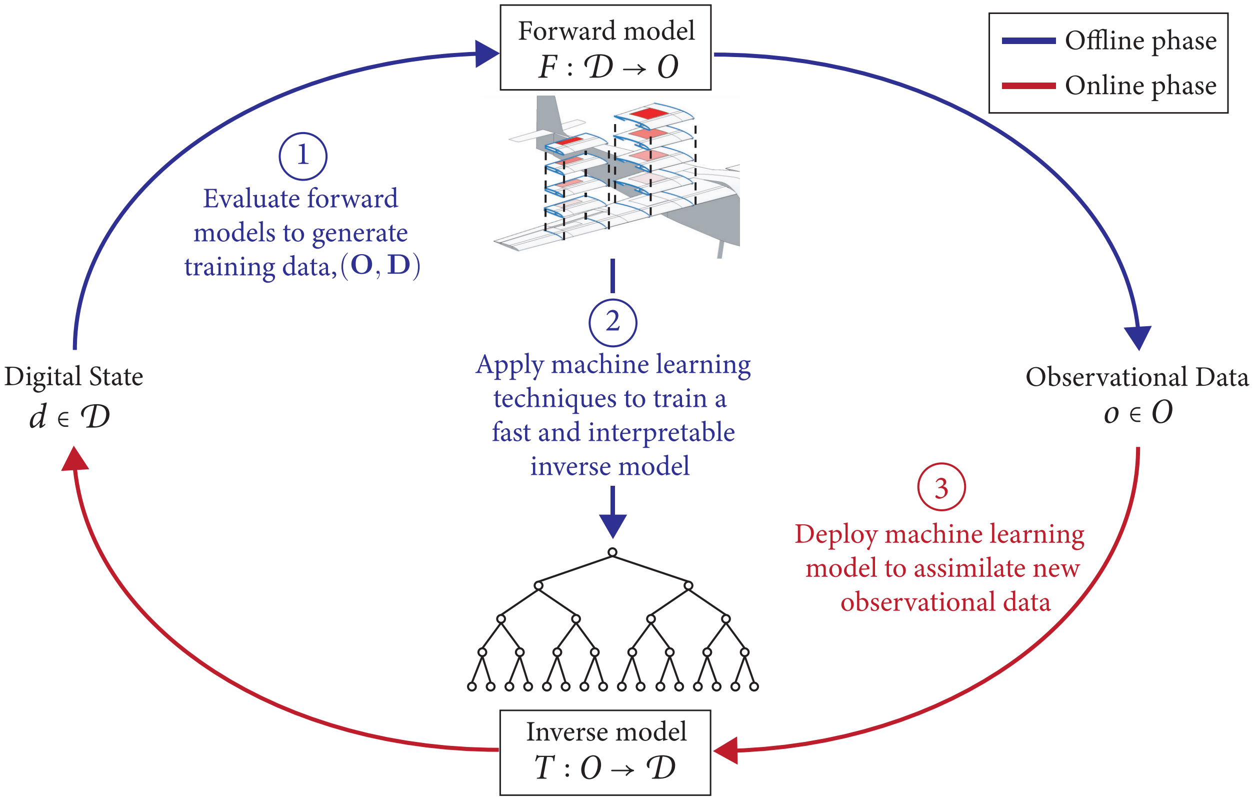

Forward model
$F : \mathcal{D} \rightarrow \mathcal{O}$
Offline phase
Online phase
1
Evaluate forward models to generate training data, $(\mathbf{O}, \mathbf{D})$
2
Apply machine learning techniques to train a fast and interpretable inverse model
Digital State
$d \in \mathcal{D}$
Observational Data
$o \in \mathcal{O}$
3
Deploy machine learning model to assimilate new observational data
Inverse model
$T : \mathcal{O} \rightarrow \mathcal{D}$

Fig. 4 An overview of the proposed approach for designing digital twin sensing strategies by combining the strengths of physics-based models and machine learning classification techniques.

physics-based predictive model. By training an OCT to minimize misclassification of this data, we are seeking to ensure that our digital twin is updated according to these same physical relationships.

The result of this approach is an OCT which defines a digital twin sensing strategy. In particular, the OCT is a computationally efficient interpretable classifier, T, that enables data data-driven adaptation of the digital twin state according to Eqn. 3. The classifier is interpretable in the sense that the result of the classification can be interpreted physically (i.e., which physical sensors and what local levels of physical strain contribute to a given classification). Moreover, the OCT can also specify (via sensor placement or sensor selection) the space of observational data, $\mathcal{O}$. In the case of a sensor placement task, the subset of sensors that appear in splits of the OCT are precisely the sensors that must be installed in order to evaluate the OCT classifier; thus, the structure of the learned OCT reveals goal-oriented sensor placement choices. Note also that the number of sensors appearing in the OCT can be controlled via the regularization term in the objective function (1), or can be explicitly constrained in the optimization problem in order to discover sparse sensing strategies. OCTs can also be used as the basis for sensor scheduling by noting that the splits in an OCT can be traversed sequentially and in prioritized order, allowing for only a subset of sensors to be queried at each timestep until a classification decision is reached. In addition, we explore in the following section how different OCTs (potentially utilizing different sensors) can be used to estimate different components of the digital state. This can be used as the basis for goal-oriented dynamic sensor scheduling.

The data for training our OCTs will be generated using predictive models. We refer to T as an *inverse model* since the observational data is dependent on the asset state. The corresponding *forward model* is defined as

$$F : \mathcal{D} \rightarrow \mathcal{O}. \tag{4}$$

This forward model maps from a given digital state to a corresponding piece of observational data. In our setting, the predictive physics-based models comprising a digital twin serve as the forward model. However, it is often the case in practice that even if the physical asset is perfectly modeled by the digital twin models with some digital state, d, the observational, o, are prone to corruption, e.g., due to sensor noise or inspection errors. We model this corruption as a random additive noise term drawn from a known noise model. An example of this is additive Gaussian noise for a sensor with known bias and covariance. It is critical to account for these noise characteristics when generating an OCT so that it can leverage this information, e.g., to learn that a given observation is typically too noisy to be reliably informative. We thus define the noisy forward mapping

$$\tilde{F} : \mathcal{D} \times \mathcal{V} \to \mathcal{O}, \tag{5}$$

where $\mathcal{V}$ is the space of possible noise values.

We generate a training dataset by sampling from this noisy forward mapping, $\tilde{F}$. To do so, we first sample from the noise-free forward mapping (4) by evaluating the predictive digital twin models to generate pairs $(o_j, d_j) \in \mathcal{O} \times \mathcal{D}, j = 1, ..., |\mathcal{D}|$, and then sample a noise vector $v^k \in \mathcal{V}$ to generate a noisy prediction of the observed quantities, namely $o_j^k = o_j + v^k$. Here the superscript k denotes the index of the noise sample. We denote by N_s the number of noisy samples we draw for each unique digital state d_j. The value of N_s depends on the allowable size of the training dataset, which in turn depends on the computational resources available for training the machine learning model.

The resulting dataset takes the form $\{(o_j^k, d_j)\}$, $j = 1, ..., |\mathcal{D}|$, $k = 1, ..., s$. As a final step, we vertically stack all of the training datapoints into a matrix representation $(\mathbf{O}, \mathbf{D})$, where $\mathbf{O} \in \mathbb{R}^{N_t \times N_o}$ and $\mathbf{D} \in \mathbb{R}^{N_t \times N_d}$. Here N_o is the dimension of the observational data, and $N_t = N_s|\mathcal{D}|$ is the total number of datapoints in the training set. As a supervised machine learning approach, the performance of a trained OCT during the online phase will depend on how well this training dataset (generated during the offline phase) represents the real-world asset operation. Specifically, we rely on the digital twin state space, $\mathcal{D}$, being a good representation of the states encountered during operation. Note that here, for simplicity, we have assumed that the $\mathcal{D}$ is discrete and finite—in practice we may have to discretize a continuous state-space and/or generate a representative sample of digital states for training. Moreover, we rely on the digital twin predictive models providing an accurate evaluation of the noisy forward mapping (5), which requires both accurate physics-based simulation capability, and an accurate model of sensor noise characteristics. Ensuring that these requirements are met through verification and validation of predictive models is a critical aspect of creating a digital twin that remains a major challenge. Our previous work specifically considered calibration and uncertainty quantification of predictive models in the digital twin context[29]. In the current work we build upon that framework by operationalizing the calibrated models for sensor design and utilization for digital twin updating. Additionally, we note that experimental or operational data in the form of measured asset data with known state, if available, could be added to augment the training dataset, while still leveraging predictive modeling for regions of the state space where models are accurate and/or data are unavailable.

IV. Application to UAV structural digital twin

This section presents a case study that serves to demonstrate the proposed approach. Section IV.A presents a fixed-wing UAV that serves as the subject of this case study. Section IV.B describes the library of physics-based models for the UAV structure which serves as the basis for its digital twin, while Section IV.C describes observational data that the digital twin has access to. In Section IV.D we describe the procedures used to generate training data comprised of predicted sensor measurements for different structural states, and also to generate OCTs based on this training data and evaluate their performance. Finally, Section IV.E explores the performance of the proposed methodology for this application.

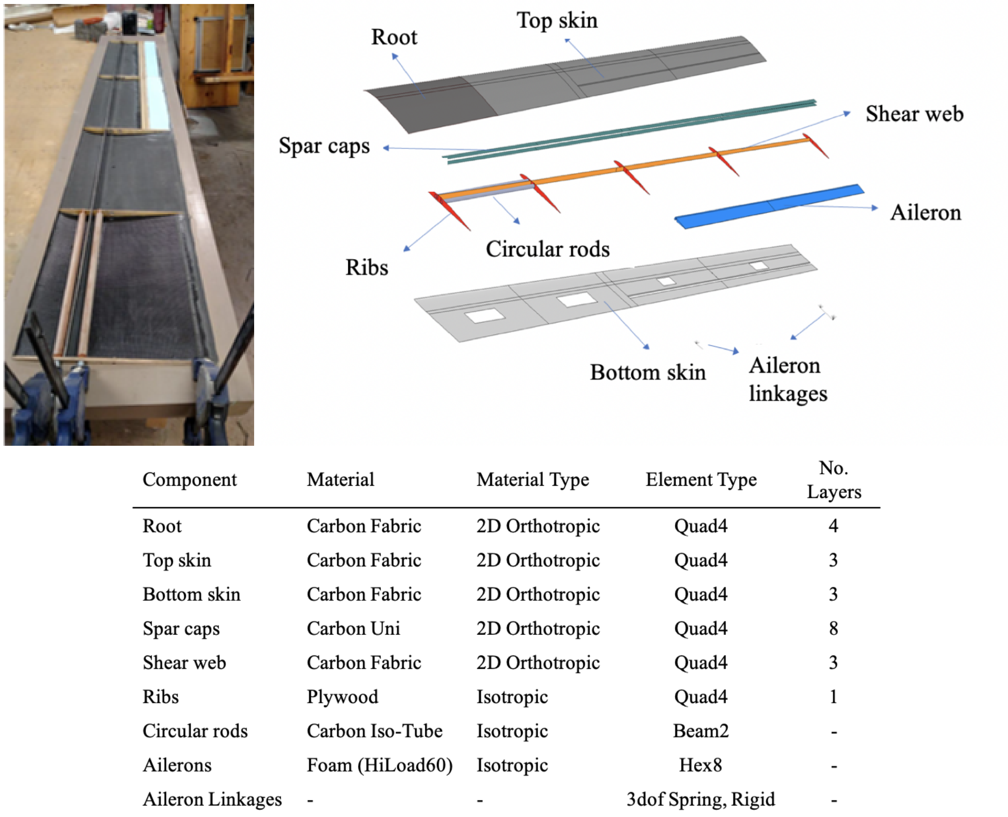
A. Physical asset: Fixed-wing unmanned aerial vehicle

Our physical asset for this research is a 12-foot wingspan, fixed-wing UAV. The fuselage is from an off-the-shelf Telemaster airplane, while the wings are custom-built with plywood ribs and a carbon fiber skin. The electric motor and avionics are also a custom installation. The wings of the aircraft are designed to be interchangeable so that the aircraft can be flown with pristine wings or wings inflicted with varying degrees of damage. This allows us to collect in-flight structural sensor data for multiple structural states, and test whether a digital twin of the aircraft is capable of dynamically detecting changes in the state using structural sensor data. We define the physical state, S, of the UAV asset to be its structural state. Although in this work we have only manufactured a single UAV asset, we target applications in which a fleet of similar UAV assets would be constructed. The physical state-space thus encompasses any conceivable structural variation between UAV assets, such as differences in geometry and material or structural properties. While many UAVs in the fleet would share a common design, variation in material properties and manufacturing processes makes no two physical assets truly identical. Furthermore, once the asset enters operation, the structural state begins to evolve over time, for example due to maintenance events, damage, or gradual degradation. It is this variability between assets, and across the asset lifecycle, that we seek to capture by creating, calibrating, and dynamically evolving a structural digital twin for each unique UAV asset.

In this case study, we consider an illustrative scenario in which the right wing of the UAV has been subjected to structural damage or degradation in flight. This change in structural state could undermine the structural integrity of the wing and reduce the maximum allowable load, thereby shrinking the flight envelope of the aircraft. A self-aware UAV should be capable of detecting and estimating the extent of this structural change, so that it can update the flight envelope and replan its mission accordingly. In this case study we enable this self-awareness through a data-driven physics-based digital twin.

B. Physics-based predictive models

The physics-based model library that serves as the foundation of the UAV digital twin in this case-study is presented in prior work[30]. We provide a high-level overview of the model library herein, and refer the reader to these works for further details. The model library is derived from a component-based reduced-order model for solving the linear elasticity equations on the full 3D UAV airframe. Figure 5 shows the internal structure of the actual UAV wing being modelled, and details the structure of the wing as represented in the computational model. Note that this model is experimentally calibrated in order to accurately match the behavior of the UAV hardware testbed described in Section IV.A, see [29] for details on the calibration process.



Component	Material	Material Type	Element Type	No. Layers
Root	Carbon Fabric	2D Orthotropic	Quad4	4
Top skin	Carbon Fabric	2D Orthotropic	Quad4	3
Bottom skin	Carbon Fabric	2D Orthotropic	Quad4	3
Spar caps	Carbon Uni	2D Orthotropic	Quad4	8
Shear web	Carbon Fabric	2D Orthotropic	Quad4	3
Ribs	Plywood	Isotropic	Quad4	1
Circular rods	Carbon Iso-Tube	Isotropic	Beam2	-
Ailerons	Foam (HiLoad60)	Isotropic	Hex8	-
Aileron Linkages	-	-	3dof Spring, Rigid	-

Fig. 5 The internal structure of the aircraft wing (actual physical wing and computational model). We use a combination of material properties and element types in order to capture the level of detail required to accurately model structural health in our digital twin.

We model an altered structural health state (e.g., caused by damage, degradation, or some other structural event) through an effective reduction in stiffness. The UAV model contains 28 so-called "defect regions" distributed across the airframe. For each defect region we introduce a scalar parameter governing the percentage reduction in material stiffness within the region. To illustrate the proposed methodology we will restrict our focus to consider only two of the defect regions on the aircraft, but note that the methodology would naturally scale to accommodate more defect regions.

The two defect regions we focus on are located on the upper surface of the right-wing. For this application we define the digital state to be two structural health parameters

$$d = [z_1, z_2] \in \mathcal{D} = \{0, 20, 40, 60, 80\} \times \{0, 20, 40, 60, 80\} \tag{6}$$

where z_1 and z_2 define the percentage reduction in material stiffness applied to defect regions 1 and 2 respectively. Here the digital state space, $\mathcal{D}$, corresponds to the 25 possible combinations of z_1 and z_2, i.e., the space of possible structural health states that the digital twin is capable of representing. This is illustrated in Figure 6.

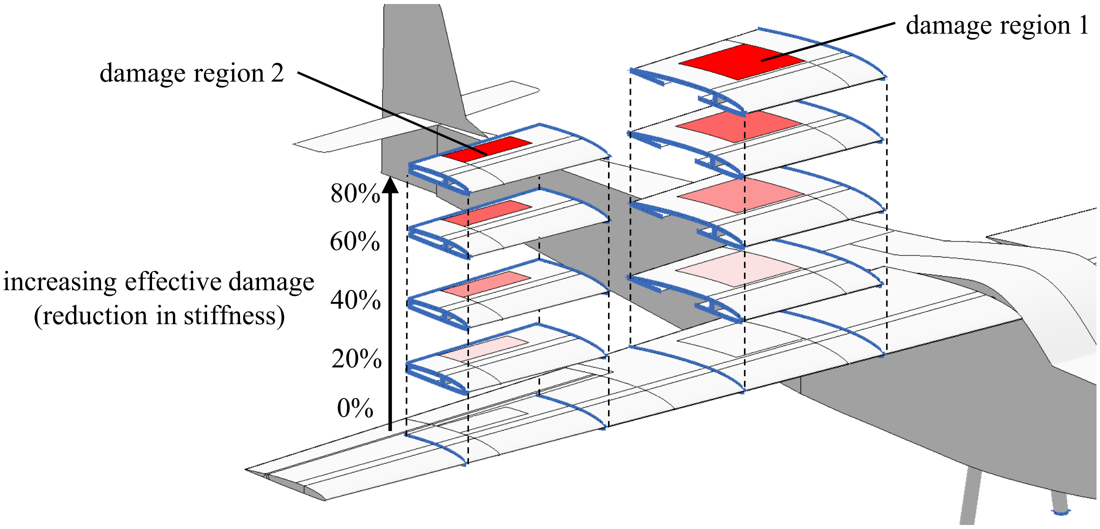


Fig. 6 An illustration of the model library used in this case study. We sample five values of the stiffness parameter in each of the defect regions (highlighted red). The model library is constructed by taking all combinations of stiffness parameters for the two components.

In addition to the structural model, we also employ an ASWING [41, 42] model of the aircraft to compute aerodynamic loads on the aircraft for steady turning maneuvers of a given load factor, L. The computed aerodynamic load is applied to the structural model in order to compute the airframe deformation corresponding to a load factor.

C. Observational data: wing-mounted strain sensors

The data-driven digital twin is enabled by using in-flight sensor data to classify the current structural state of the UAV into one of the $|\mathcal{D}| = 25$ states capable of being represented by the digital twin. The sensors we consider are uniaxial strain gauges mounted on the top surface of the right wing. Since the data from each strain gauge takes the form of a positive real number, our space of possible observational data is defined as $O = \mathbb{R}^{N_s}$, where N_s is the number of sensors under consideration.

We consider as a baseline the sensor suite installed on the physical UAV asset, as shown in Figure **??**. In particular, we consider 24 uniaxial strain gauges distributed in two span-wise rows on either side of the main spar between 25% and 75% span. Figure 7 shows a schematic of the right wing with sensor locations. For this case-study, we exclude sensors 17, 18, 23 and 24 (shown as gray circles in Fig. 7) since these sensors are located in the defect regions. Thus, for this case study $N_s = 20$ unless stated otherwise.

D. Training and evaluation procedure

We here describe the procedure we adopt for generating and evaluating optimal trees for this case study. In this scenario our observations are noisy strain measurements at strain gauge locations. These measurements take the form

$$\boldsymbol{\epsilon}(d, u) + \mathbf{v}^k, \tag{7}$$

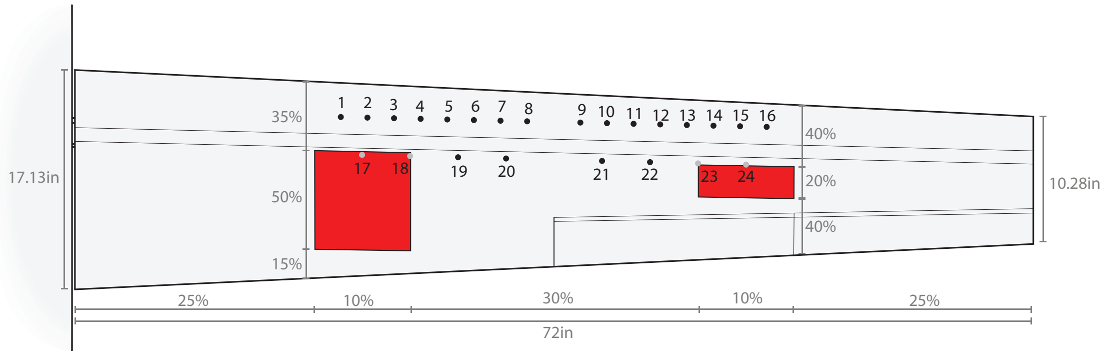


Fig. 7 Schematic of the UAV wing. The two defect regions we consider in this case-study are highlighted in red. Uniaxial strain gauges are shown as circles and are referred to by their integer label. Sensors 17, 18, 23 and 24 are excluded from this case-study.

where $\boldsymbol{\epsilon}$ is a vector where each element is the true strain at one of the N_o strain gauge locations. The strain depends on the load factor, $u = L$, (ratio of lift force to aircraft weight), and the UAV structural state, as represented by the digital twin computational models with digital state d_j. In this illustration we model the sensor noise as zero-mean white noise, $\mathbf{v}^k$, so that

$$\mathbf{v}^k \sim \mathcal{N}(\mathbf{0}, 1000\mathbb{I}), \tag{8}$$

x where $\mathcal{N}$ denotes a Gaussian distribution and $\mathbb{I} \in \mathbb{R}^{N_o \times N_o}$ is an identity matrix.

Our structural model of the UAV is based on linear elasticity, so the strain predicted by the model is linear with respect to the applied load factor. Since the load factor is typically known based on the aircraft kinematics, we normalize measurements by the load factor and define the observed data to be a vector of load-normalized strains of form

$$o_j^k = \frac{\boldsymbol{\epsilon}(d_j, L) + \mathbf{v}^k}{L}. \tag{9}$$

We generate training data of this form for every model in the component-based reduced-order model library, $z_j, j = 1, ..., |\mathcal{D}| = 25$, by drawing samples from the noisy forward mapping, $\tilde{F}$ (5). The procedure for drawing samples is as follows: We first compute the true strain $\boldsymbol{\epsilon}(d, u)$ using the predictive computational models described in Sec. IV.B, for each digital state, d_j. We then draw a random noise sample $\mathbf{v}_k$, according to (8), and add this to the computed strain. Finally, we normalize the noisy strain by the load factor to give a feature vector o_j^k, consisting of N_o noisy, load-normalized strain measurements. The final step is repeated for $N_s = 100$ noise samples $k = 1, ..., s$. Note that generating the training data in this way is a many-query task, requiring $|\mathcal{D}|$ evaluations of the UAV structural model. This is a one-time, offline procedure, and in our case is accelerated by the use of reduced-order models in the model library.

This process results in a dataset containing a total of $N_t = 2500$ datapoints. Each datapoint is of the form $(\mathbf{o}_j^k, [z_1, z_2]_j)$. The feature vector, $\mathbf{o}_j^k$ is a prediction of the N_o noisy load-normalized strain measurements corresponding to the structural state represented by the two parameters, z_1 and z_2, which can take one of 5 discrete values corresponding to 0%, 20%, ..., 80% reduction in stiffness respectively.

Recall that our goal is to use load-normalized strain data in order to classify the UAV structural health into a structural state represented by one of the $|\mathcal{D}| = 25$ states represented in the model library. Directly classifying the UAV structural health into one of the library models requires a tree of at least depth five, so that it has at least 25 possible labels (one for each leaf in the tree). Since each model has two parameters, z_1, and z_2, we choose to train a separate optimal tree classifier for each parameter. This is so that each tree requires only 5 possible labels (a depth of at least three). This is simply to make the resulting trees more interpretable since the total state d_j can be easily inferred from the two parameters. In particular, note that we can recover a single tree for directly classifying d_j by simply appending the tree for classifying z_2 to every leaf of the tree for classifying z_1. The resulting tree predicts both z_1 and z_2 for a given input, which corresponds uniquely to state d_j. To this end we define the classifier output to be either z_1 or z_2.

To generate optimal trees we use a Julia implementation of the local-search algorithm described in Section III.C, provided in the Interpretable AI software package.[43] This implementation allows one to specify a number of parameters and constraints to be used in the optimization problem (1). We set the complexity trade-off parameter, κ, to zero and the minimum number of training points in each leaf of the tree to one. To control the complexity of the resulting tree we explicitly constrain the maximum depth and the maximum number of features appearing in each split, which we refer to as the split complexity. This approach allows us to explore the tradeoff between classification accuracy and complexity in a controlled way, by generating the tree with optimal classification accuracy for a given constrained complexity. Note that in practice these parameters do not need to be set explicitly and can instead by found using standard cross-validation techniques.

To evaluate the performance of an optimal tree classifier, we follow standard machine learning practice and hold out 30% of the data (750 datapoints) as a testing set. We feed each test datapoint into the classifiers to determine its estimated structural state, given by the parameter z_1 or z_2, and compare this with the structural state that was used to generate that test point. Recall that here we are using classification as a computationally efficient alternative to a continuous regression. In this context, error metrics that focus on some form of misclassification rate would be sensitive to the choice of discretization used for z_1 and z_2. For this reason, we instead adopt a more regression oriented error metric and quantify the performance of trained optimal tree classifiers by computing the Mean Absolute Error (MAE) in z_1 or z_2, i.e., the error between the true percentage reduction in stiffness, and the reduction in stiffness estimated by the optimal tree classifier.

E. Results

In this section we apply the proposed methodology to the UAV structural digital twin application, and explore various aspects of its performance. First, we demonstrate how OCTs can be interpreted as defining different sensing strategies for the UAV digital twin, given a fixed set of sensors. Next, we explore the trade-off between interpretability and classification performance of OCTs for this application. Finally, we demonstrate how the methodology can be used for optimal sensor placement, and how this offers performance benefits over an ad-hoc sensor placement approach.

1. Sensor selection and data assimilation for a baseline sensor architecture

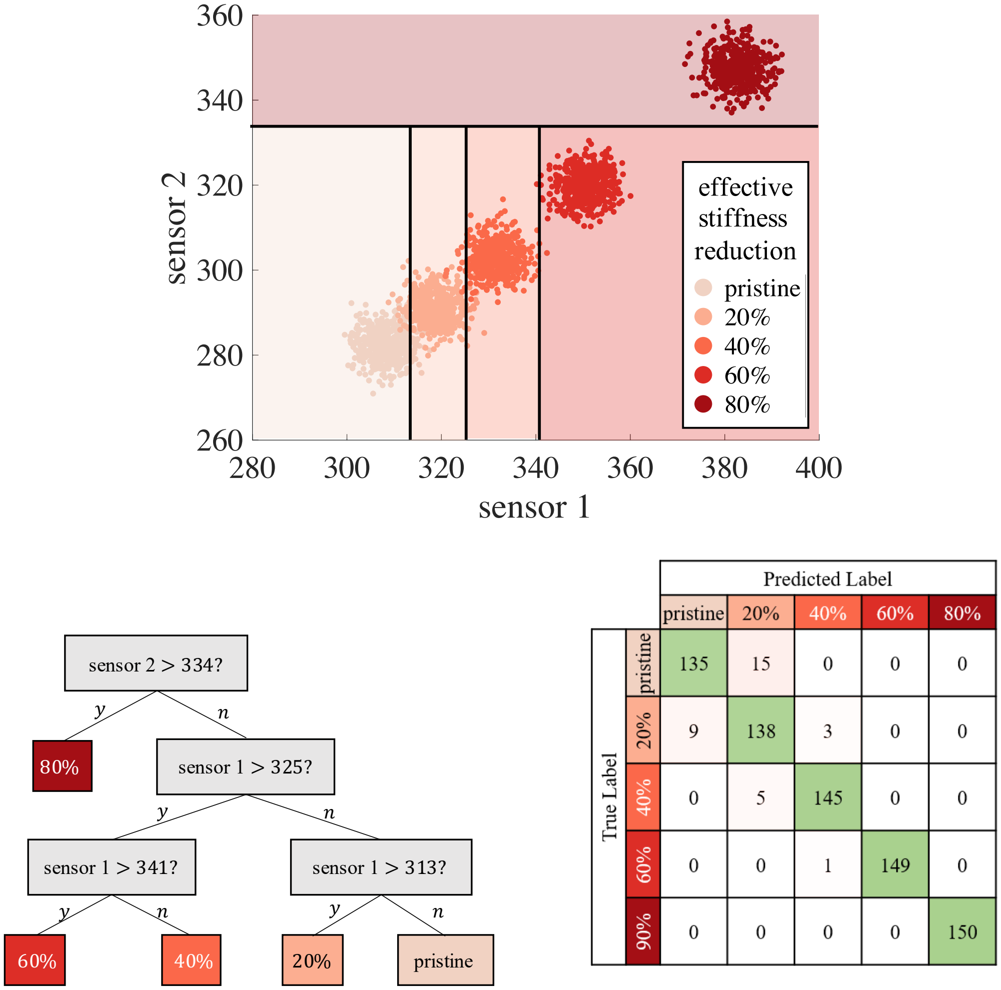
We begin by demonstrating how the OCTs correspond to a partitioning of the feature space, in this case, the space of all possible structural sensor measurements from the UAV, and how the structure of the tree corresponds to a sparse selection of sensors that are required to classify the structural state. To this end, we start by considering the baseline strain gauge locations as shown in Figure 7, but seek sparse sensing strategies by constraining the optimal trees to have a maximum depth of three and to utilize, at most, two sensors. This is so that the resulting trees are highly efficient and more interpretable, and so that the corresponding partitioning of the feature space can be easily visualized on a two-dimensional subspace of O (each dimension corresponding to one of the two chosen sensors).

We first focus on classifying the first digital state parameter, z_1 using axis-aligned splits. A resulting OCT trained for this task utilizes sensors 1 and 2, and is shown in Figure 8. Also shown is the corresponding partitioning of the feature space and the confusion matrix.

The MAE for this classifier is 0.65 on the training set, which corresponds to 57 out of the 1750 training datapoints being misclassified (a misclassification rate of 3.3%). The MAE is higher for the test set at 0.88. This shows that sensors 1 and 2 are sufficient for accurately classifying the reduction in stiffness in defect region 1. This choice of sensors agrees with intuition as they are both close to the defect region. Note however, that no explicit information about proximity is present in the training data. Instead, the optimization problem for generating the optimal tree finds that sensors 1 and 2 are the most informative for this classification task, and thus develops a data assimilation strategy based on these two sensors.

Figure 8 shows that strain increases with z_1 equally for both sensor 1 and 2, suggesting that axis-aligned splits might not be suited to this problem. Figure 9 shows the results obtained using a hyperplane splits involving sensors 1 and 2. In this case the MAE improves to 0.22 on the training set, which corresponds to 20 out of the 1750 training datapoints being misclassified (a misclassification rate of 1.1%). The MAE on the test set also improves to 0.4. This highlights how increasing the number of sensors appearing in each split can improve the classification performance of the tree, at the cost of making the classifier more complex, and thus less interpretable. This trade-off is further explored in Subsection IV.E.2. Note that in this case, the classifiers found using both parallel and hyperplane splits are not unique: This training data exhibits a relatively high signal-to-noise ratio, resulting in good separation between data clusters corresponding to each digital twin state. This allows the optimization to move or rotate certain splits slightly without affecting the misclassification rate of the resulting classifier.

Classifying the second structural health parameter, z_2, is more challenging since the second defect region is smaller



		Predicted Label				
		pristine	20%	40%	60%	80%
True Label	pristine	135	15	0	0	0
	20%	9	138	3	0	0
	40%	0	5	145	0	0
	60%	0	0	1	149	0
	90%	0	0	0	0	150

Fig. 8 An OCT for computing the digital state parameter z_1 using axis-aligned splits. Top: Partitioning of the space of strain measurements. Bottom left: Corresponding OCT for classifying the value of z_1. Bottom right: Corresponding confusion matrix.

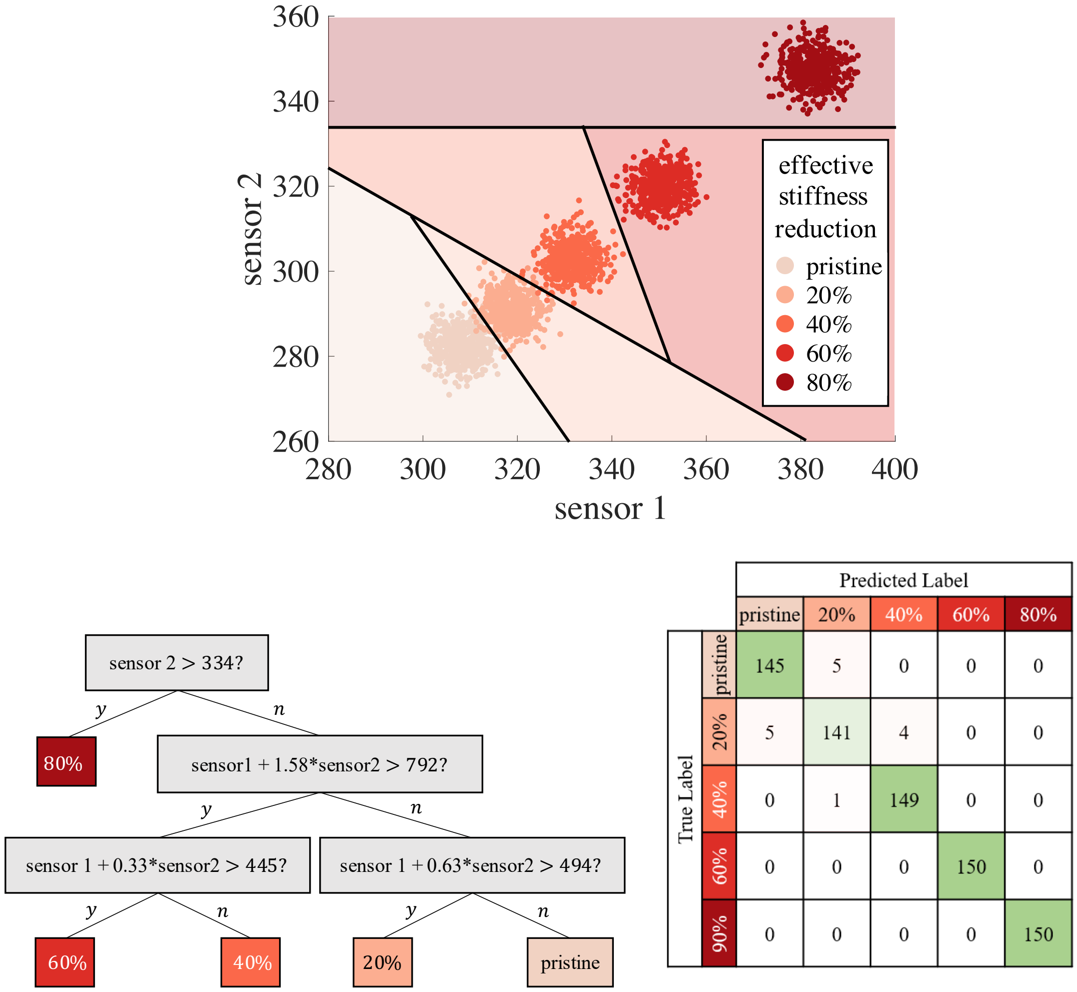
than the first, and nearer to the wing tip, thus having a lesser effect of wing deflection and the resulting strain field. The OCT trained to estimate z_2 is shown in Figure 10.

In this case, the MAE is 19.44 on the training set and 20.7 on the test set. This corresponds to a high misclassification rate: 60.1% and 64.4% respectively. This high misclassification rate can also be seen in the confusion matrix in Figure 10. This high error is due to the fact that the datapoints corresponding to different parameter values are not easily separable using just two sensors (as seen in Figure 10). Improving this error requires either a more complex tree or improving our sensing capability. We explore the latter option further in the Subsection IV.E.3.

2. Performance vs. complexity tradeoff

In this section we explore the tradeoff between classification performance and complexity of the optimal trees used in the UAV digital twin. Figure 11 shows the MAE on the training and test sets for optimal trees trained to classify either z_1 and z_1 with varying tree depth and split complexities. When it comes to classifying z_1 (Figure 11, top row), we see that with a tree depth of at least four and a split complexity of at least two the optimal tree classifier is able to fit the training data exactly. Performance on the test set is marginally lower in all cases, but is still strong.

On the other hand, for optimal trees trained for the more difficult task of classifying z_2 (Figure 11, bottom row) we see that the ability of the optimal tree to fit the training data increases with both depth and sparsity. Note the difference in axis scaling when compared with Figure 11. Here the optimal trees achieve an MAE of 7.76 with a tree of depth six and



		Predicted Label				
		pristine	20%	40%	60%	80%
True Label	pristine	145	5	0	0	0
	20%	5	141	4	0	0
	40%	0	1	149	0	0
	60%	0	0	0	150	0
	90%	0	0	0	0	150

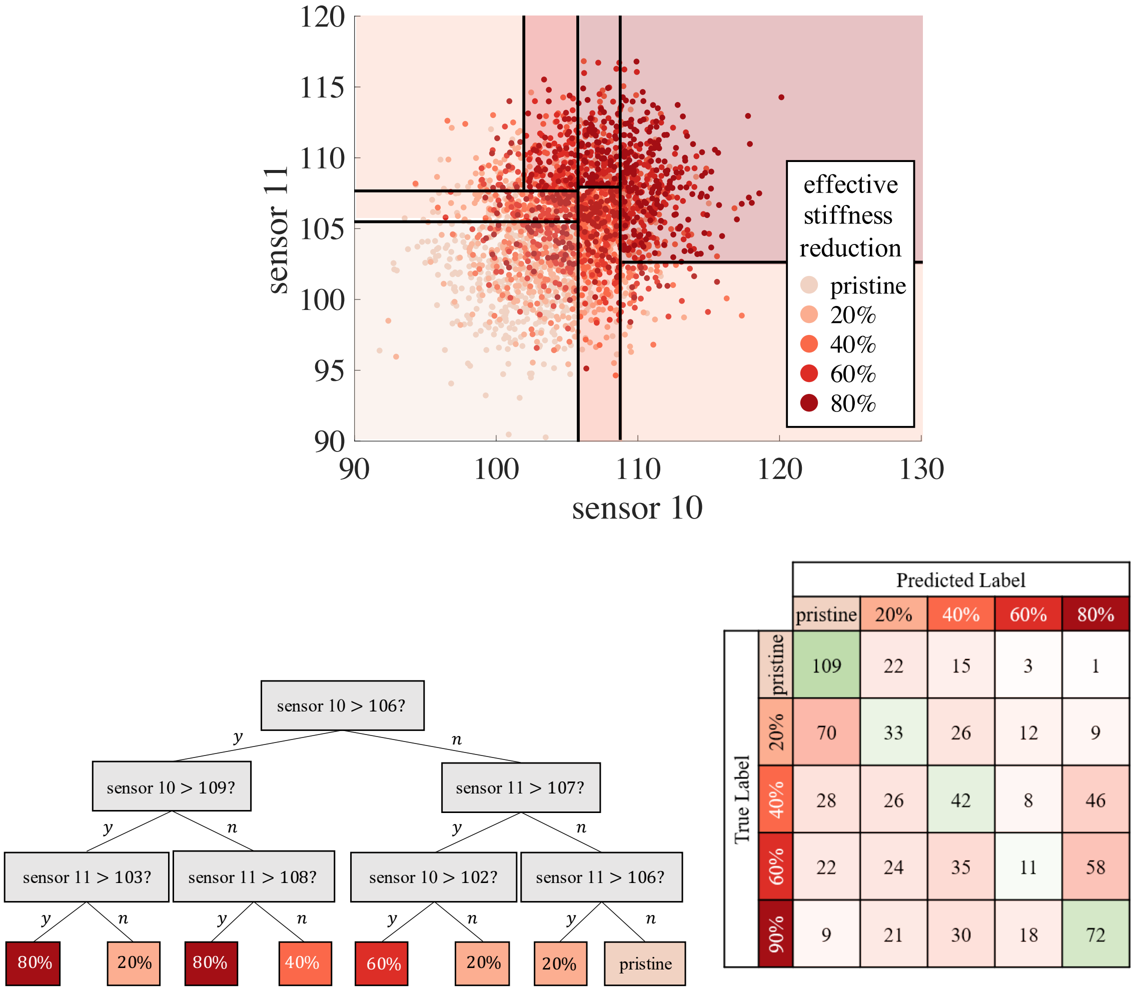
Fig. 9 An OCT for computing the digital state parameter z_1 using hyperplane splits. Top: Partitioning of the space of strain measurements. Bottom left: Corresponding OCT for classifying the value of z_1. Bottom right: Corresponding confusion matrix.

unlimited split complexity, a reduction in error of roughly 54% compared with the depth three tree using axis-aligned splits. The out-of-sample performance on the testing set improves uniformly with split complexity, but does not improve uniformly with depth. We see that the out-of-sample performance is best at depth four, suggesting that the depth five and six trees are over-fitting the training data.

3. Improved sensor design via optimal sensor placement

In the previous analysis, we utilized the baseline sensor suite currently installed on the hardware testbed. The strain gauges installed on the aircraft are not positioned optimally for the classification task we consider in this case study. Instead, the sensor placement was determined through an ad hoc analysis whereby the pristine wing finite element model was solved, and regions of high strain were identified. Note that the optimal sensor placements would be areas where the strain field is most *sensitive* to changes in the structural state, which are not necessarily the regions where the strain magnitude is *highest*.

A more principled approach to positioning the sensors involves leveraging optimal trees to directly determine which sensor locations are most informative for updating the digital twin. We do this by first determining a large set of feasible sensor locations and then training optimal classification trees using these candidate sensors as input features. Recall that the local search method we adopt for training classification trees scales linearly with the number of input features, allowing us to efficiently explore a number of candidate sensors that is significantly larger than the final number of



True Label	Predicted Label: pristine	20%	40%	60%	80%
pristine	109	22	15	3	1
20%	70	33	26	12	9
40%	28	26	42	8	46
60%	22	24	35	11	58
90%	9	21	30	18	72

Fig. 10 An OCT for computing the model parameter z_2. Top: Partitioning of the feature space (the space of strain measurements) using axis-aligned splits. Bottom left: The corresponding OCT for classifying the value of z_2. Bottom right: The corresponding confusion matrix.

sensors to be installed. To illustrate this, we consider a set of 58 candidate sensors in addition to the 24 sensors already installed. The candidate sensors are arranged in four spanwise rows along the wing, as shown in Figure 12. As with the baseline sensor suite, we exclude sensor locations that occur within a defect region, giving a total of $N_s = 67$ possible sensor locations, and thus $N_s = 67$ features to inform classification in the optimal tree. We adapt the methodology described in Section IV.D to generate a dataset that includes all 67 features corresponding to the complete set of candidate sensors. Figure 13 shows the MAE for OCTs of varying depth trained using the candidate sensor locations compared with using the already installed sensors. To focus the comparison on the positioning of sensors rather than the quantity of available sensors, we limit the total number of sensors appearing in each tree by setting the hyperplane complexity to four.

We see that the optimal trees determine that adding a subset of the candidate sensors would lead to improved classification performance. For example, in the depth three case, the optimal tree trained using the candidate set utilizes seven sensors that are already installed on the aircraft, and an additional six of the candidate sensors. This set of sensors results in an MAE of 9.89 as opposed to 11.64 for the fixed sensor set. We observe similar trends for tree depths four and five.

These results show how the optimal trees methodology can scale to a large number of sensors, and how utilizing the optimal trees methodology earlier in the development of a digital twin can serve as a principled approach to sensor placement that can lead to improved accuracy of the digital twin and fewer required sensors.

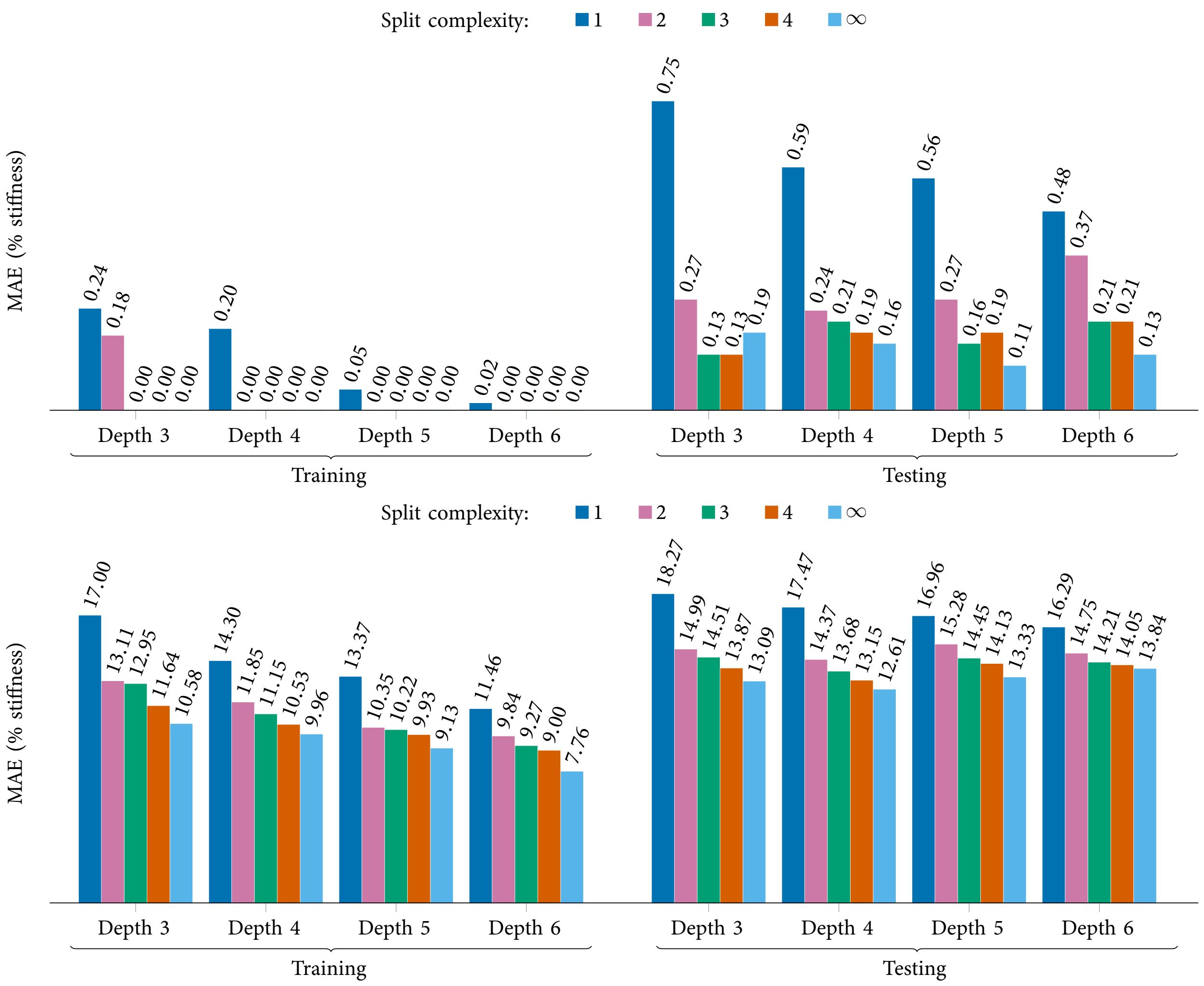


Fig. 11 Plots of classification performance (MAE) versus tree depth and split complexity. Left: performance evaluated on the training set. Right: performance evaluated on the test set. Top row: Classifiers for parameter z_1. Bottom row: Classifiers for parameter z_2.

V. Conclusion

This work has presented an approach for designing digital twin sensing strategies via a combination of predictive modeling and machine learning. Physics-based predictive models facilitate the generation of a rich dataset containing predictions of observational data for a wide range of digital states. This dataset is used to train an OCT that provides data-driven estimates of the digital state, i.e., the configuration of physics-based models within the digital twin that best matches the data. Using this classifier online enables computationally efficient digital twin data assimilation. In addition, the OCT structure can be used to inform sensor placement and sensor scheduling decisions. This approach has been demonstrated using a case study in which a fixed-wing UAV uses structural sensors to detect degradation on its wings. The case study employs a Gaussian noise model for illustration, but in practice, characterization of the sensors on the system at hand is an essential step that must be incorporated into the OCT training process. Deployment of the approach with data from the actual physical system is an important next step.

A limitation of the approach is that it assumes that the digital state space is known and fixed a priori. In practice, it is infeasible to define a digital state space that encompasses all possible current and future physical asset states. How to define the digital state space in a principled way, and how to construct the predictive physics-based models associated with each digital state efficiently, remains an open area of research. Alternatively, an adaptive digital state space could enable the digital twin to expand or refine its modeling capabilities on-the-fly. In this case, the proposed methodology

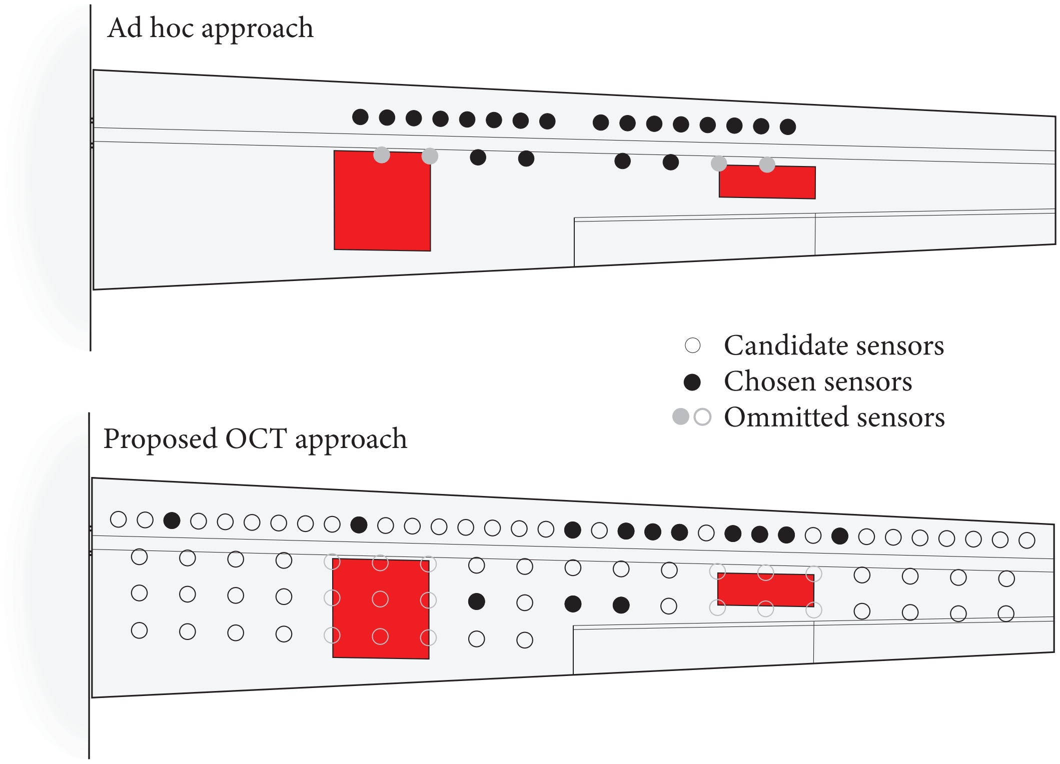


Fig. 12 Sensor placement results for the UAV application. Top: The baseline sensor suite selected using an ad hoc approach. Bottom: Sensor locations chosen via the proposed OCT approach.

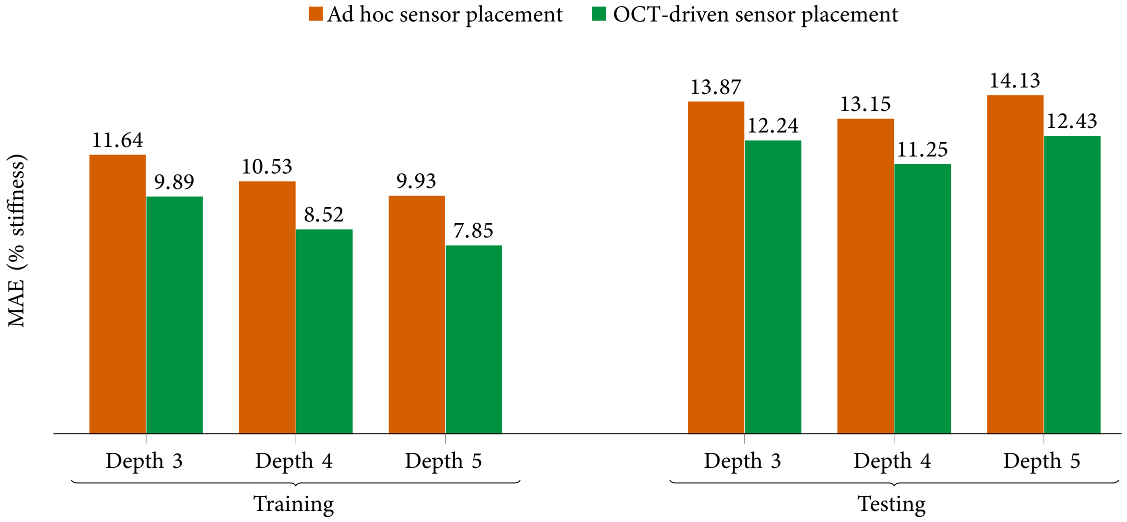


Fig. 13 Plot of classification performance (MAE) versus tree depth for optimal trees trained to classify z_2 using either the existing fixed set of sensors, or the larger set of candidate sensor locations.

would need to be adapted to incorporate retraining or adaptation of the OCTs as new states are added to the space. A further limitation is that as the dimension of the digital state space becomes large, the amount of training data required to train the classifier may become prohibitive. This again points to the need for adaptive training strategies.

Future research into the proposed methodology could explore variations in the optimization problem used to generate the optimal classification trees, for example, by incorporating a strict sensor budget either globally or at each time step.

Another area for future study is the robustness of optimal trees to sensor anomalies and failures, and the investigation of whether exploiting the interpretable nature of these classifiers could enable one to detect and mitigate such failures.

Acknowledgments

This work was supported in part by DOE grant DE-SC0021239 and AFOSR grants FA9550-18-1-0023, FA9550-21-1-0084 and FA9550-21-1-0089. The authors gratefully acknowledge C. Kays and other collaborators at Aurora Flight Sciences for construction of the physical UAV asset. The authors also acknowledge their collaborators D.J. Knezevic, D.B.P. Huynh, and M. Tran at Akselos for their work on the development of the reduced-order structural model of the UAV used in this work. Finally, the authors thank J. Dunn and others at Interpretable AI for the use of their software.

References

[1] Grieves, M., and Vickers, J., *Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems*, Springer International Publishing, 2017, pp. 85–113. doi:10.1007/978-3-319-38756-7_4, URL https://doi.org/10.1007/978-3-319-38756-7_4.

[2] Rasheed, A., San, O., and Kvamsdal, T., "Digital Twin: Values, Challenges and Enablers From a Modeling Perspective," *IEEE Access*, Vol. 8, 2020, pp. 21980–22012. doi:10.1109/ACCESS.2020.2970143.

[3] Niederer, S. A., Sacks, M. S., Girolami, M., and Willcox, K., "Scaling digital twins from the artisanal to the industrial," *Nature Computational Science*, Vol. 1, No. 5, 2021, pp. 313–320.

[4] AIAA Digital Engineering Integration Committee, "Digital Twin: Definition & Value," , 2020. URL https://www.aia-aerospace.org/report/digital-twin-paper/.

[5] Tuegel, E. J., Ingraffea, A. R., Eason, T. G., and Spottswood, S. M., "Reengineering Aircraft Structural Life Prediction Using a Digital Twin," *International Journal of Aerospace Engineering*, Vol. 2011, 2011. doi:10.1155/2011/154798.

[6] Glaessgen, E., and Stargel, D., *The Digital Twin Paradigm for Future NASA and U.S. Air Force Vehicles*, 2012. doi: 10.2514/6.2012-1818, URL https://arc.aiaa.org/doi/abs/10.2514/6.2012-1818.

[7] Li, C., Mahadevan, S., Ling, Y., Choze, S., and Wang, L., "Dynamic Bayesian Network for Aircraft Wing Health Monitoring Digital Twin," *AIAA Journal*, Vol. 55, No. 3, 2017, pp. 930–941. doi:10.2514/1.J055201, URL https://doi.org/10.2514/1.J055201.

[8] Podskarbi, M., and Knezevic, D. J., "Digital Twin for Operations - Present Applications and Future Digital Thread," 2020. doi:10.4043/30553-MS, URL https://doi.org/10.4043/30553-MS.

[9] Kraft, J., and Kuntzagk, S., "Engine Fleet-Management: The Use of Digital Twins From a MRO Perspective," 2017. doi:10.1115/GT2017-63336, URL https://doi.org/10.1115/GT2017-63336.

[10] Reifsnider, K., and Majumdar, P., "Multiphysics Stimulated Simulation Digital Twin Methods for Fleet Management," *54th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, 2013. doi:10.2514/6.2013-1578, URL https://arc.aiaa.org/doi/abs/10.2514/6.2013-1578.

[11] Bruynseels, K., Santoni de Sio, F., and van den Hoven, J., "Digital twins in health care: Ethical implications of an emerging engineering paradigm," *Frontiers in Genetics*, Vol. 9, 2018, p. 31. doi:10.3389/fgene.2018.00031.

[12] Rivera, L. F., Jiménez, M., Angara, P., Villegas, N. M., Tamura, G., and Müller, H. A., "Towards Continuous Monitoring in Personalized Healthcare through Digital Twins," *Proceedings of the 29th Annual International Conference on Computer Science and Software Engineering*, IBM Corp., USA, 2019, p. 329–335.

[13] Barricelli, B. R., Casiraghi, E., Gliozzo, J., Petrini, A., and Valtolina, S., "Human Digital Twin for Fitness Management," *IEEE Access*, Vol. 8, 2020, pp. 26637–26664. doi:10.1109/ACCESS.2020.2971576.

[14] Hernandez-Boussard, T., Macklin, P., Greenspan, E. J., Gryshuk, A. L., Stahlberg, E., Syeda-Mahmood, T., and Shmulevich, I., "Digital twins for predictive oncology will be a paradigm shift for precision cancer care," *Nature Medicine*, 2021, pp. 1–2.

[15] Yu, H., Miao, C., Leung, C., and White, T. J., "Towards AI-powered personalization in MOOC learning," *npj Science of Learning*, Vol. 2, No. 1, 2017, pp. 1–5. doi:10.1038/s41539-017-0016-3.

[16] Mohammadi, N., and Taylor, J. E., "Smart city digital twins," *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2017, pp. 1–5. doi:10.1109/SSCI.2017.8285439.

[17] Kennedy, M. C., and O'Hagan, A., "Bayesian calibration of computer models," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, Vol. 63, No. 3, 2001, pp. 425–464. doi:10.1111/1467-9868.00294, URL `https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/1467-9868.00294`.

[18] Zhang, Y., de Visser, C. C., and Chu, Q. P., "Aircraft Damage Identification and Classification for Database-Driven Online Flight-Envelope Prediction," *Journal of Guidance, Control, and Dynamics*, Vol. 41, No. 2, 2018, pp. 449–460. doi:10.2514/1.G002866, URL `https://doi.org/10.2514/1.G002866`.

[19] Zakrajsek, A. J., and Mall, S., "The Development and use of a Digital Twin Model for Tire Touchdown Health Monitoring," *58th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, 2017. doi:10.2514/6.2017-0863, URL `https://arc.aiaa.org/doi/abs/10.2514/6.2017-0863`.

[20] Zhao, W., Gupta, A., Regan, C. D., Miglani, J., Kapania, R. K., and Seiler, P. J., "Component data assisted finite element model updating of composite flying-wing aircraft using multi-level optimization," *Aerospace Science and Technology*, Vol. 95, 2019, p. 105486. doi:10.1016/j.ast.2019.105486, URL `https://www.sciencedirect.com/science/article/pii/S1270963819304572`.

[21] Chinesta, F., Cueto, E., Abisset-Chavanne, E., Duval, J. L., and El Khaldi, F., "Virtual, digital and hybrid twins: a new paradigm in data-based engineering and engineered data," *Archives of computational methods in engineering*, Vol. 27, No. 1, 2020, pp. 105–134. doi:10.1007/s11831-018-9301-4.

[22] Moya, B., Badías, A., Alfaro, I., Chinesta, F., and Cueto, E., "Digital twins that learn and correct themselves," *International Journal for Numerical Methods in Engineering*, 2020. doi:10.1002/nme.6535, URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.6535`.

[23] Yucesan, Y. A., and Viana, F., "A hybrid model for main bearing fatigue prognosis based on physics and machine learning," *AIAA Scitech 2020 Forum*, 2020. doi:10.2514/6.2020-1412, URL `https://arc.aiaa.org/doi/abs/10.2514/6.2020-1412`.

[24] Kapteyn, M. G., Knezevic, D. J., and Willcox, K., "Toward predictive digital twins via component-based reduced-order models and interpretable machine learning," *AIAA Scitech 2020 Forum*, 2020, p. 0418.

[25] Bertsimas, D., and Dunn, J., "Optimal classification trees," *Machine Learning*, Vol. 106, No. 7, 2017, pp. 1039–1082. doi:10.1007/s10994-017-5633-9.

[26] Bertsimas, D., and Dunn, J., *Machine Learning Under a Modern Optimization Lens*, Dynamic Ideas LLC, 2019.

[27] Brunton, B. W., Brunton, S. L., Proctor, J. L., and Kutz, J. N., "Sparse sensor placement optimization for classification," *SIAM Journal on Applied Mathematics*, Vol. 76, No. 5, 2016, pp. 2099–2122.

[28] Mainini, L., and Willcox, K. E., "Data to decisions: Real-time structural assessment from sparse measurements affected by uncertainty," *Computers & Structures*, Vol. 182, 2017, pp. 296–312. doi:10.1016/j.compstruc.2016.12.007, URL `https://www.sciencedirect.com/science/article/pii/S0045794916308197`.

[29] Kapteyn, M. G., Pretorius, J. V. R., and Willcox, K. E., "A probabilistic graphical model foundation for enabling predictive digital twins at scale," *Nature Computational Science*, Vol. 1, No. 5, 2021. doi:10.1038/s43588-021-00069-0, URL `https://www.nature.com/articles/s43588-021-00069-0`.

[30] Kapteyn, M. G., Knezevic, D. J., Huynh, D. B. P., Tran, M., and Willcox, K. E., "Data-driven physics-based digital twins via a library of component-based reduced-order models," *International Journal for Numerical Methods in Engineering*, 2020. doi:10.1002/nme.6423, URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.6423`, special Issue on Digital Twins.

[31] Quinlan, J. R., "Induction of Decision Trees," *Machine Learning*, Vol. 1, No. 1, 1986, pp. 81–106. doi:10.1007/BF00116251.

[32] Rokach, L., and Maimon, O., *Decision Trees*, Springer US, Boston, MA, 2005, pp. 165–192. doi:10.1007/0-387-25465-X_9, URL `https://doi.org/10.1007/0-387-25465-X_9`.

[33] Fisher, R. A., "The use of multiple measurements in taxonomic problems," *Annals of Eugenics*, Vol. 7, No. 2, 1936, pp. 179–188. doi:10.1111/j.1469-1809.1936.tb02137.x, URL `https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1469-1809.1936.tb02137.x`.

[34] Dua, D., and Graff, C., "UCI Machine Learning Repository," , 2017. URL `http://archive.ics.uci.edu/ml`.

[35] Bertsimas, D., Dunn, J., and Wang, Y., "Near-optimal Nonlinear Regression Trees," *Operations Research Letters*, Vol. 49, No. 2, 2021, pp. 201–206. doi:https://doi.org/10.1016/j.orl.2021.01.002, URL `https://www.sciencedirect.com/science/article/pii/S0167637721000031`.

[36] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," , 2019. URL `http://www.gurobi.com`.

[37] Dunn, J. W., "Optimal Trees for Prediction and Prescription," Ph.D. thesis, Massachusetts Institute of Technology, 2018. URL `http://dspace.mit.edu/handle/1721.1/7582`.

[38] Breiman, L., Friedman, J., Olshen, R., and Stone, C., *Classification And Regression Trees*, 2017. doi:10.1201/9781315139470.

[39] Breiman, L., "Random Forests," *Machine Learning*, Vol. 45, No. 1, 2001, pp. 5–32. doi:10.1023/A:1010933404324.

[40] Friedman, J. H., "Greedy function approximation: A gradient boosting machine." *The Annals of Statistics*, Vol. 29, No. 5, 2001, pp. 1189 – 1232. doi:10.1214/aos/1013203451, URL `https://doi.org/10.1214/aos/1013203451`.

[41] Drela, M., "Integrated simulation model for preliminary aerodynamic, structural, and control-law design of aircraft," *40th Structures, Structural Dynamics, and Materials Conference and Exhibit*, 1999. doi:10.2514/6.1999-1394, URL `https://arc.aiaa.org/doi/abs/10.2514/6.1999-1394`.

[42] Drela, M., "ASWING 5.99 Technical Description—Steady Formulation," , March 2015. URL `http://web.mit.edu/drela/Public/web/aswing/`.

[43] Interpretable AI, LLC, "Interpretable AI Documentation," , 2020. URL `https://www.interpretable.ai`.