Received November 18, 2020, accepted December 14, 2020, date of publication January 18, 2021, date of current version January 26, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3052130

Selecting Informative Data Samples for Model Learning Through Symbolic Regression

ERIK DERNER[1,2], (Student Member, IEEE), JIŘÍ KUBALÍK[1], AND ROBERT BABUŠKA[1,3], (Member, IEEE)

[1]Czech Institute of Informatics, Robotics, and Cybernetics, Czech Technical University in Prague, 16000 Prague, Czech Republic
[2]Department of Control Engineering, Faculty of Electrical Engineering, Czech Technical University in Prague, 16627 Prague, Czech Republic
[3]Department of Cognitive Robotics, Delft University of Technology, 2628 CD Delft, The Netherlands

Corresponding author: Erik Derner (erik.derner@cvut.cz)

This work was supported in part by the European Regional Development Fund through the project Robotics for Industry 4.0 under Grant CZ.02.1.01/0.0/0.0/15_003/0000470, and in part by the Grant Agency of the Czech Technical University in Prague under Grant SGS19/174/OHK3/3T/13.

ABSTRACT Continual model learning for nonlinear dynamic systems, such as autonomous robots, presents several challenges. First, it tends to be computationally expensive as the amount of data collected by the robot quickly grows in time. Second, the model accuracy is impaired when data from repetitive motions prevail in the training set and outweigh scarcer samples that also capture interesting properties of the system. It is not known in advance which samples will be useful for model learning. Therefore, effective methods need to be employed to select informative training samples from the continuous data stream collected by the robot. Existing literature does not give any guidelines as to which of the available sample-selection methods are suitable for such a task. In this paper, we compare five sample-selection methods, including a novel method using the model prediction error. We integrate these methods into a model learning framework based on symbolic regression, which allows for learning accurate models in the form of analytic equations. Unlike the currently popular data-hungry deep learning methods, symbolic regression is able to build models even from very small training data sets. We demonstrate the approach on two real robots: the TurtleBot mobile robot and the Parrot Bebop drone. The results show that an accurate model can be constructed even from training sets as small as 24 samples. Informed sample-selection techniques based on prediction error and model variance clearly outperform uninformed methods, such as sequential or random selection.

INDEX TERMS Machine learning, system identification, robot control, genetic algorithms, symbolic regression.

I. INTRODUCTION

To effectively control nonlinear dynamic systems, such as autonomous robots, one needs accurate models. These models can be learned and adapted by using data samples that the robot continuously collects during its deployment. As the amount of such data quickly grows with time, using all the collected samples for model learning soon becomes computationally infeasible, and a subset of data must be selected. However, not all data samples are equally important, and it is not known a priori which samples will be useful and which not. This problem is compounded by the presence of data samples from repetitive motions, which are typical for most tasks in robotics. Such data do not contain any additional information, and without precautions, they outweigh the relatively small amount of other informative samples.

Data samples for model learning can be chosen in an uninformed way or in an informed way. Most prominent among the uninformed approaches are the recursive methods [1], [2] used in classical system identification. They process data sequentially, use every sample only once to update the model parameters and then throw it away. This makes them data-inefficient and unable to address the issue with repetitive samples. Another widely used uninformed approach is the random selection of training samples [3], [4], which also does not solve the problem with repetitive samples.

Informed methods usually work with a set of models. A typical representative of this class is the variance approach [5]. The key idea of this method is that the most informative sample is the one that causes the largest disagreement among the models found. Related methods have also been

The associate editor coordinating the review of this manuscript and approving it for publication was Hiram Ponce.

developed in the field of *active learning* [3], known mostly for applications in classification [6]–[8], but also applied to regression [9]–[11]. Active learning starts with a small number of labeled training samples and then iteratively requests labels for additional samples. The labels are obtained from an oracle, often a human expert, which makes labeling expensive. In model learning, the labels are the measured system outputs. The problem here is that the output for an arbitrary sample cannot be obtained from a real dynamic system, as the system would have to be brought to the required state, which is often undesired or impossible.

An alternative to the approaches that require a set of models are methods that do not rely on the learned models' outputs. A representative of these methods is the problem domain coverage [3]. This approach iteratively adds new samples to evenly cover the problem domain, thus saving the computational costs of learning multiple models.

In addition to the informed methods based on the model variance [5] and on the domain coverage [3], we propose a novel approach based on the model prediction error. In contrast to the variance method, the new sample added to the training set is the one with the highest error averaged over the current set of models. The motivation is to deal with cases when the set of models yields a low variance for a given sample, but the models' outputs on that sample are all wrong. Such a sample would be disregarded by the variance method, though it is clearly worth adding to the training data set. This happens, for example, in case the function (model) sought has some unexpected property on a small part of its domain, which has not been covered by the samples from the previous iterations. Contrary to the variance method, the prediction error method can also work with a single model.

To compare the above approaches to sample selection, we introduce a framework using symbolic regression (SR) for model learning. SR has proven to be suitable for modeling nonlinear system dynamics even from very small data sets [12]. The advantage of using SR is that it constructs parsimonious models in the form of analytic equations, which facilitates their use within other algorithms. Symbolic regression allows to optionally incorporate prior knowledge in the model construction process by specifying the set of elementary functions that can be used to build the analytic models. In addition to its data efficiency, SR also requires fewer parameters to build an accurate model when compared to alternative methods such as deep neural networks [4], [13]–[15]. As SR can be time-consuming for large data sets, selecting a suitable small training set makes it very well usable in practice.

This paper makes the following two main contributions:

- We present a comparative study of five methods for selecting data samples from a larger sample collection recorded during the robot deployment. Such a comparison has been so far missing in the literature. Three informed and two uninformed methods are evaluated within the SR framework on data both from a simulated and a real mobile robot TurtleBot 2, and on data from a real drone Parrot Bebop 2.
- A new sample-selection method is introduced. It is based on adding samples that yield the largest model prediction error. The practical merit of the proposed method is demonstrated in an experiment with the real mobile robot.

The rest of the paper is organized as follows. The model learning framework is described in Section II. Sections III and IV present the experimental results and Section V concludes the paper.

II. METHODS

Section II-A first defines the nonlinear dynamic model considered and introduces the theoretical background of symbolic regression. The model learning procedure is explained in Section II-B and the sample-selection methods evaluated in this paper are described in Section II-C.

A. NONLINEAR DYNAMIC SYSTEM MODEL

The dynamic system model is described in discrete time by the following nonlinear difference equation:

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) \tag{1}$$

with n-dimensional state $\mathbf{x}_k = (x_k^1, x_k^2, \ldots, x_k^n)^\top$ and m-dimensional input $\mathbf{u}_k = (u_k^1, u_k^2, \ldots, u_k^m)^\top$, where k denotes the discrete time step. While the actual process can be stochastic (e.g., when the sensor readings are corrupted by noise), in this paper, we construct a deterministic model.

For model learning, we define the model $\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)$ as a vector of models $f^j(\mathbf{x}_k, \mathbf{u}_k)$, each producing a prediction of a single state variable x_{k+1}^j, with $j = 1, \ldots, n$:

$$\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) = \Big(f^1(\mathbf{x}_k, \mathbf{u}_k), f^2(\mathbf{x}_k, \mathbf{u}_k), \ldots, f^n(\mathbf{x}_k, \mathbf{u}_k)\Big)^\top . \tag{2}$$

In the sequel, we drop the superscripts to simplify the notation. The generic term $f(\mathbf{x}_k, \mathbf{u}_k)$ corresponds to a model of a single state variable, while the model of the whole system is denoted by $\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)$. Similarly, x_k refers to a single generic state variable, whereas $\mathbf{x}_k$ represents the full state vector.

To find a concise model of the nonlinear system dynamics, we use a variant of SR called Single Node Genetic Programming (SNGP) [16]. It forms the model $f(\mathbf{x}_k, \mathbf{u}_k)$ for each state variable as a linear combination of evolved nonlinear functions $f_i(\mathbf{x}_k, \mathbf{u}_k)$:

$$f(\mathbf{x}_k, \mathbf{u}_k) = \beta_0 + \sum_{i=1}^{n_f} \beta_i f_i(\mathbf{x}_k, \mathbf{u}_k). \tag{3}$$

The SNGP algorithm builds the functions $f_i(\mathbf{x}_k, \mathbf{u}_k)$, called features, from a user-defined set of elementary functions $\mathcal{F}$. The features are represented as directed acyclic graphs and evolved using standard evolutionary operations such as mutation. The function set can be broad to let SR choose the appropriate functions, but the user can also specify a narrower set of elementary functions to speed up the evolution by utilizing prior knowledge about the system. The evolution is driven by the minimization of the mean-square error calculated over the training data set. The coefficients β are estimated by least squares. To avoid over-fitting, the complexity of the regression model is limited by two parameters: the number of features n_f and the maximum feature depth d.

We assume that the states are measured, but the method also applies to input–output models of the form $\mathbf{y}_{k+1} = \mathbf{g}(\mathbf{y}_k, \mathbf{y}_{k-1}, \ldots, \mathbf{u}_k, \mathbf{u}_{k-1}, \ldots)$, where the state is represented by the vector of past inputs and outputs [17].

B. MODEL LEARNING FRAMEWORK

During its deployment, the system (robot) continuously collects samples in the form $\mathbf{s}_k = (\mathbf{x}_k, \mathbf{u}_k, \mathbf{x}_{k+1})^\top$. The data samples are stored in a buffer from which a subset of samples for training is selected. To evaluate the performance of the method and its ability to generalize, a small portion of the collected samples is diverted to the test set instead of becoming a part of the buffer. There are different ways how the test samples can be selected, e.g., by periodically or randomly choosing new samples for the test set to maintain a user-specified ratio between the buffer and test set size.

SR is run periodically on the training data set selected from the buffer to find the state transition function of the system. As outlined in Section II-A, symbolic regression constructs a model for each state variable individually. In the following text, we describe the learning procedure for a generic state variable. Sample selection is also performed per state variable, i.e., each variable has its own instance of the training set and the buffer.

An overview of the method is presented in Algorithm 1. The algorithm starts with an initial training data set composed of a small number of samples n_0. There are various ways how to choose the initial samples. For example, they can be chosen randomly among the samples available in the buffer. In this paper, we apply a sequential approach, starting with the first n_0 samples in the buffer.

The method builds the models iteratively, where by an iteration, we denote the process of constructing n_r analytic models and choosing a set of n_s samples from the buffer to be added to the training data set. A small value of n_s yields fine-grained sample selection and will be used when aiming at small but highly informative training data sets on which SR can be run often and with low computational costs per run. Larger values of n_s allow the training set size to grow faster, where SR will be run less often, but with higher computational costs per run.

In each iteration, symbolic regression runs in n_r identically configured instances to find models fitting the training data. Since evolution is guided by a distinct sequence of random numbers in each of the runs r, we obtain n_r different analytic models $f_r(\mathbf{x}_k, \mathbf{u}_k)$. The model f^* with the lowest root-mean-square error (RMSE) of the one-step-ahead prediction on the test set is chosen as the final model for that iteration. The set of n_r models also serves to determine the informative samples, as described in Sections II-C1 and II-C3.

The iterative process terminates once a given stopping criterion is met or once the maximum number of iterations n_i is reached. For example, the stopping criterion can be based on a threshold on the error measures or on the performance of the system on a given control task.

Algorithm 1 Model Learning With Sample Selection

```
Input: sample-selection method, $Buffer$, $TestSet$, $n_0$, $n_s$, $n_i$
$i \leftarrow 0$
$TrainingSet \leftarrow S_{n_0}$ (first $n_0$ samples in $Buffer$)
$Buffer \leftarrow Buffer \setminus S_{n_0}$
repeat
    $i \leftarrow i + 1$
    for each state variable do
        run $n_r$ instances of SR to construct models $f_r$
        $f^* \leftarrow f_r$ with the lowest RMSE on $TestSet$
        $S \leftarrow n_s$ samples from $Buffer$,
            chosen by the sample-selection method
        $TrainingSet \leftarrow TrainingSet \cup S$
        $Buffer \leftarrow Buffer \setminus S$
    end for
until $i = n_i$ or termination condition on model quality is met
```

C. SAMPLE-SELECTION METHODS

Sample selection is important to efficiently construct accurate models. The following text presents three informed sample-selection methods, followed by two uninformed methods used as a baseline for the performance analysis.

1) MAXIMUM VARIANCE

A common state-of-the-art informed sample-selection method is based on the maximum variance between the model outputs [3], [5], [18]. For a given training set, n_r models are generated and the outputs of these models are calculated for all data samples in the buffer. The data samples with the highest variance in model outputs are added to the training set. The method is based on the hypothesis that the samples with the highest variance come from a subset of the problem domain that is not sufficiently represented in the current training set. Including such samples is expected to improve the model consistency and accuracy.

2) MAXIMUM OUTPUT DOMAIN COVERAGE

In this approach, the training set is constructed by iteratively adding samples from the buffer to cover the output domain as well as possible. For each sample in the buffer, the method calculates its distance in the output space to all samples in the current training set and stores the minimum of these distances. The buffer sample with the largest minimum distance is added to the training set.

A similar approach has been used in [3] for the input domain. The approach to cover the output domain instead of the input domain is advantageous because it circumvents the issues connected to the normalization of the input space components. Unlike the maximum variance approach, the maximum output domain coverage method does not need any models to be built for sample selection and therefore, it is computationally less demanding.

3) MAXIMUM PREDICTION ERROR (PERMIT)

In addition to the two previous informed sample-selection methods, we propose a novel method that selects the samples on which the analytic models yield on average the largest error. In the sequel, we will refer to our method by its acronym PERMIT (Prediction ERror method for Model ImprovemenT).

The selection procedure is performed for each state variable individually. The models $f_r(\mathbf{x}_\ell, \mathbf{u}_\ell)$, $r = 1, 2, \ldots, n_r$, are used to calculate the output for all samples $\mathbf{s}_\ell$ in the buffer. We select the sample $\mathbf{s}_{\ell^*}$ that yields the largest prediction error averaged over the set of n_r models:

$$\ell^* = \underset{\ell \in \{1,\ldots,N\}}{\arg\max} \frac{1}{n_r} \sum_{r=1}^{n_r} \left(f_r(\mathbf{x}_\ell, \mathbf{u}_\ell) - z_\ell\right)^2, \tag{4}$$

where N is the current buffer size and $n_r \geq 1$. Recall that each sample $\mathbf{s}_\ell$ has the form $(\mathbf{x}_k, \mathbf{u}_k, \mathbf{x}_{k+1})^\top$. The term z_ℓ in (4) refers to the component of $\mathbf{x}_{k+1}$ corresponding to the modeled variable.

In contrast to the variance method, which requires a set of models to calculate the variance, PERMIT can select new training samples using only a single model. However, averaging over a set of models improves the method's robustness.

4) SEQUENTIAL ADDITION

A common uninformed sample-selection approach is to build the model from samples added in the order as they are logged [2]. The most straightforward implementation of this method is using the queue data structure. During the operation of the system, the recorded data samples are added at the tail of the queue. New samples for model learning are taken from the queue head. Therefore, the data in the buffer are processed in the first-in, first-out (FIFO) manner.

5) RANDOM APPROACH

This approach selects the samples at random. The training samples are drawn from the buffer with a uniform probability. This method has been used as a reference also in [3]. While the method works well for buffers with a majority of informative samples, its performance degrades if the available data samples have been recorded mostly from repetitive motions and rich data form only a small portion of the collected data set.

D. COMPUTATIONAL COMPLEXITY

The computational complexity of symbolic regression grows linearly with the number of samples. The complexity of the sample-selection method grows linearly with the buffer size and also linearly with the number of parameters in the analytic model.

To provide an idea on the actual computation time, we have measured the time needed to finish a single SR run for different sample sizes. We have used a standard laptop computer with a CPU Intel Core i7-4610M (3.00 GHz) and 16 GB of RAM, running the computation on a single core. Consider a problem with a three-dimensional state and a two-dimensional input, such as the mobile robot described in Section III. The time used by SR to find a model of the system is approximately 28 seconds for a training set of 20 samples, 75 seconds for a training set of 100 samples, and 240 seconds for a training set of 500 samples. The sample-selection method itself takes a negligible amount of time (< 50 ms) for $n_r = 10$ models and buffers containing thousands of samples. The advantage of using a suitable sample-selection method to reduce the number of training samples is therefore substantial.

E. DISCUSSION AND LIMITATIONS

We consider a scenario in which the system (robot) performs a given task, e.g., transporting objects between specified locations, and we can not alter its behavior in any way. Often, this leads to unevenly covered state and input domains. The majority of samples span a small subset of the state and input domain and only a small portion of samples are spread across other parts of the domain. A different situation would arise if we had full control over the system operation and we could design a task yielding samples evenly and densely covering the state and input space. These two scenarios are closely related to the classical exploration-exploitation dilemma. While the first scenario corresponds to exploitation, the second one represents exploration. We focus in this paper on the first scenario, which is characteristic for the operation in regular task execution. The strength of the informative sample selection manifests in particular in that case, as the choice of the right samples is crucial for the modeling performance.

The proposed method is designed to work with any system and does not make any prior assumptions. The only requirement is that the dynamics are excited during the task execution in at least a small portion of the collected data samples (approx. 10–20 %, depending on the task). This ensures a training data set sufficiently rich in informative samples capturing the properties of the system. It is generally satisfied for highly dynamic tasks (rapid motions of robots), but it may not be satisfied for stationary tasks (e.g., a quadcopter hovering above a fixed location), in the absence of external disturbances.

Finally, we assume that the buffer does not contain a large amount of corrupted data (outliers). We have empirically evaluated that the method is robust to a small number of erroneous records (less than 10 %) present in the training set.

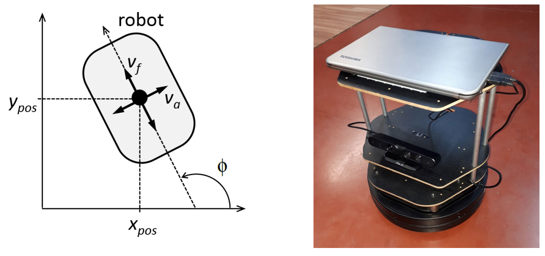


FIGURE 1. Mobile robot: a) schematic, b) photo of the TurtleBot used in the experiments.

III. MOBILE ROBOT EXPERIMENTS

We have chosen a mobile robot as a suitable benchmark for our method. We have carried out experiments both in simulations and with a real mobile robot TurtleBot 2.

A. SYSTEM DESCRIPTION

We consider a two-wheeled mobile robot shown in Fig. 1. Its model is described by the state vector $\mathbf{x} = (x_{pos}, y_{pos}, \phi)^\top$, where x_{pos} and y_{pos} are the position coordinates of the robot and ϕ is its heading. The forward velocity v_f and the angular velocity v_a are the control inputs, forming the input vector $\mathbf{u} = (v_f, v_a)^\top$.

The theoretical continuous-time model of the mobile robot is:

$$\begin{aligned} \dot{x}_{pos} &= v_f \cos(\phi), \\ \dot{y}_{pos} &= v_f \sin(\phi), \\ \dot{\phi} &= v_a, \end{aligned} \tag{5}$$

neglecting the dynamics caused by the robot's inertia and actuators. This model is only used for simulations in Section III-E. In our experiments, the inputs are limited to the domain $v_f \in [0, 0.3]\,\mathrm{m}\cdot\mathrm{s}^{-1}$ and $v_a \in [-1, 1]\,\mathrm{rad}\cdot\mathrm{s}^{-1}$, which substantiates the use of the sampling period $T_s = 0.2\,\mathrm{s}$.

B. DATA COLLECTION

The data sets both in simulations and in the experiments with the real robot are composed of short sequences such as moving forward with the maximal forward velocity, turning on the spot with the maximal angular velocity, turning in a circle with the maximal forward and angular velocity, or waiting on the spot for a new command. Approximately 20 % of the data are sequences with random inputs within the domain for v_f and v_a. The first two thirds in each of these sequences form the buffer (a total of 500 samples), while the last third enters the test set (250 samples).

The data samples have the following form:

$$\mathbf{s}_k = (x_{pos,k}, y_{pos,k}, \phi_k, v_{f,k}, v_{a,k}, x_{pos,k+1}, y_{pos,k+1}, \phi_{k+1}), \tag{6}$$

where k denotes the time step, see (1). Odometry measurements were used to record the samples in the experiments with the real robot.

C. MODEL LEARNING

The model learning algorithm starts with a training data set of $n_0 = 5$ first samples in the buffer. The initial training data set is identical for all three state variables. The aim is to select a small subset of training data that will capture the robot's dynamics as accurately as possible. In each iteration, $n_r = 50$ analytic models are constructed. At the end of each iteration, we add $n_s = 1$ sample to the training set. We limit the number of iterations to $n_i = 50$. We have deliberately chosen extremely low values of the parameters n_0 and n_s to show that even very small data sets can serve to build accurate models of the robot. In practice, higher values of these parameters could be used to reduce the number of initial iterations in which we do not yet expect the models to be sufficiently accurate.

The analytic models were constructed by using SNGP with up to $n_f = 10$ features having a maximum depth of $d = 7$. The set of elementary functions for symbolic regression was $\mathcal{F} = \{\times, +, -, \sin, \cos, \text{square}, \text{cube}\}$.

We have evaluated the five sample-selection methods described in Section II-C. We use the one-step-ahead RMSE calculated on the test data set to evaluate the quality of the analytic models found by SR in each iteration. It compares the output of the model $f_r(\mathbf{x}_k, \mathbf{u}_k)$ with the known next state component for the given variable x_{k+1}, which is stored with the test sample. Median RMSE values are calculated over all n_r models in each iteration. Due to the randomness factor in SR, the sample-selection process is stochastic. Therefore, the results shown represent one particular realization of a stochastic process. Using 50 repetitions of the experiment, we have empirically validated that these results are representative for the performance of the methods.

D. CONTROL TASK

In addition to the RMSE measure, we evaluate the performance of the analytic models on a control task. The robot has to reach the reference (goal) state $\mathbf{x}_r$ from a given initial state $\mathbf{x}_0$ as fast as possible. Fuzzy V-iteration is employed to find an approximation of the V-function in a model-based reinforcement learning (RL) scheme. The description of the RL algorithm is beyond the scope of this paper; for details, please refer to [19]. We set the discount factor γ to 0.99. The reward function is equal to zero if the robot is within $\pm 0.01\,\mathrm{m}$ in x_{pos}, $\pm 0.01\,\mathrm{m}$ in y_{pos}, and $\pm 0.02\,\mathrm{rad}$ in ϕ from the reference state $\mathbf{x}_r$. Otherwise, the reward is -1. This leads to minimum-time optimal control from an initial pose to the specified neighborhood of the goal pose.

An analytic model f^* with the lowest RMSE on the test data set is selected for each variable in each iteration to be used within RL. For the control task, we limit the state domain to $x_{pos} \in [0, 1]\,\mathrm{m}$, $y_{pos} \in [0, 1]\,\mathrm{m}$, and $\phi \in (-\pi, \pi]\,\mathrm{rad}$. We set the reference state to the center: $\mathbf{x}_r = (x_{pos,r}, y_{pos,r}, \phi_r)^\top = (0.5, 0.5, 0)^\top$. The control input is selected from a set of 11 values spanning evenly the range $v_f \in [0, 0.2]\,\mathrm{m}\cdot\mathrm{s}^{-1}$ and from 21 values spanning evenly the range $v_a \in [-0.5, 0.5]\,\mathrm{rad}\cdot\mathrm{s}^{-1}$.

TABLE 1. Comparison of the sample-selection methods on All variables in the experiments with the mobile robot. The RMSE median and the RMSE spread Are averaged over All iterations of the sample-selection procedure. The RMSE spread Is calculated as the difference between the maximum and minimum error among the models in each iteration.

	Selection method	Simulation			Real robot		
		x_{pos} [m]	y_{pos} [m]	ϕ [rad]	x_{pos} [m]	y_{pos} [m]	ϕ [rad]
Average RMSE median	variance	$2.52 \cdot 10^1$	$3.16 \cdot 10^1$	$9.58 \cdot 10^0$	$4.13 \cdot 10^1$	$3.82 \cdot 10^0$	$3.21 \cdot 10^1$
	coverage	$5.71 \cdot 10^1$	$4.04 \cdot 10^1$	$1.36 \cdot 10^1$	$7.07 \cdot 10^1$	$4.59 \cdot 10^0$	$7.14 \cdot 10^1$
	PERMIT	$2.52 \cdot 10^1$	$3.16 \cdot 10^1$	$8.68 \cdot 10^0$	$4.09 \cdot 10^1$	$3.86 \cdot 10^0$	$3.51 \cdot 10^1$
	sequential	$6.21 \cdot 10^2$	$5.30 \cdot 10^2$	$5.92 \cdot 10^2$	$1.65 \cdot 10^3$	$5.27 \cdot 10^2$	$1.93 \cdot 10^3$
	random	$2.02 \cdot 10^2$	$8.14 \cdot 10^1$	$8.37 \cdot 10^1$	$1.76 \cdot 10^2$	$6.73 \cdot 10^1$	$1.25 \cdot 10^2$
Average RMSE spread	variance	$9.18 \cdot 10^5$	$9.32 \cdot 10^{15}$	$6.38 \cdot 10^5$	$3.38 \cdot 10^{15}$	$2.02 \cdot 10^{13}$	$5.51 \cdot 10^{12}$
	coverage	$1.90 \cdot 10^7$	$4.26 \cdot 10^{21}$	$2.34 \cdot 10^{27}$	$1.59 \cdot 10^{11}$	$6.17 \cdot 10^{10}$	$8.08 \cdot 10^{33}$
	PERMIT	$9.18 \cdot 10^5$	$9.32 \cdot 10^{15}$	$6.27 \cdot 10^5$	$3.38 \cdot 10^{15}$	$2.02 \cdot 10^{13}$	$5.51 \cdot 10^{12}$
	sequential	$4.80 \cdot 10^{35}$	$1.31 \cdot 10^{28}$	$1.13 \cdot 10^{30}$	$2.55 \cdot 10^{33}$	$1.18 \cdot 10^{58}$	$6.86 \cdot 10^{30}$
	random	$7.59 \cdot 10^{21}$	$3.71 \cdot 10^{14}$	$1.58 \cdot 10^{11}$	$8.33 \cdot 10^{13}$	$2.53 \cdot 10^{21}$	$6.70 \cdot 10^{26}$

We introduce two control performance measures. The mean distance from the reference state is calculated as the mean of the Euclidean distances between $(x_{pos,k}, y_{pos,k})^\top$ and $(x_{pos,r}, y_{pos,r})^\top$ for all steps $k = 1, 2, \ldots, n_k$. Furthermore, it is averaged over all experiments executed from various initial states. This measure captures the underlying goal to transit from the initial state to the reference state as efficiently as possible. In addition, the mean error in the final state is calculated as the mean Euclidean distance between the final state $\mathbf{x}_{n_k}$ and the reference state $\mathbf{x}_r$, averaged over experiments from all initial states. Note that the latter measure includes the robot heading ϕ and the error for ϕ is normalized to the same range as for x_{pos} and y_{pos}.

To evaluate the control task, we start the simulation experiments from a grid of 64 initial states spanning the state domain. The duration of each simulation is 20 seconds, which corresponds to $n_k = 100$ steps (excluding the initial state). In the case of the real robot, the control task is executed every five iterations from four initial states $(0, 0, 0)^\top$, $(0.1, 0.9, 0)^\top$, $(0.8, 0.8, 0)^\top$, and $(0.8, 0.2, 0)^\top$.

E. SIMULATION RESULTS

At first, we have performed a set of simulation experiments. We simulate the mobile robot by applying the fourth-order Runge-Kutta integration method [20] to the equations of motion (5). Note that the simulation model does not constitute a part of our method; it only serves to generate the data samples.

Table 1 summarizes the results in two quantitative measures: the mean of the median RMSE and the mean difference between the RMSE of the best and the worst model in each iteration. The median RMSE is calculated over $n_r = 50$ models in each iteration and both measures are averaged over all iterations. These measures allow for evaluating how accurate models can the sample-selection methods construct from small data sets. All the sample-selection methods would converge to models of the same accuracy when using all samples from the buffer.

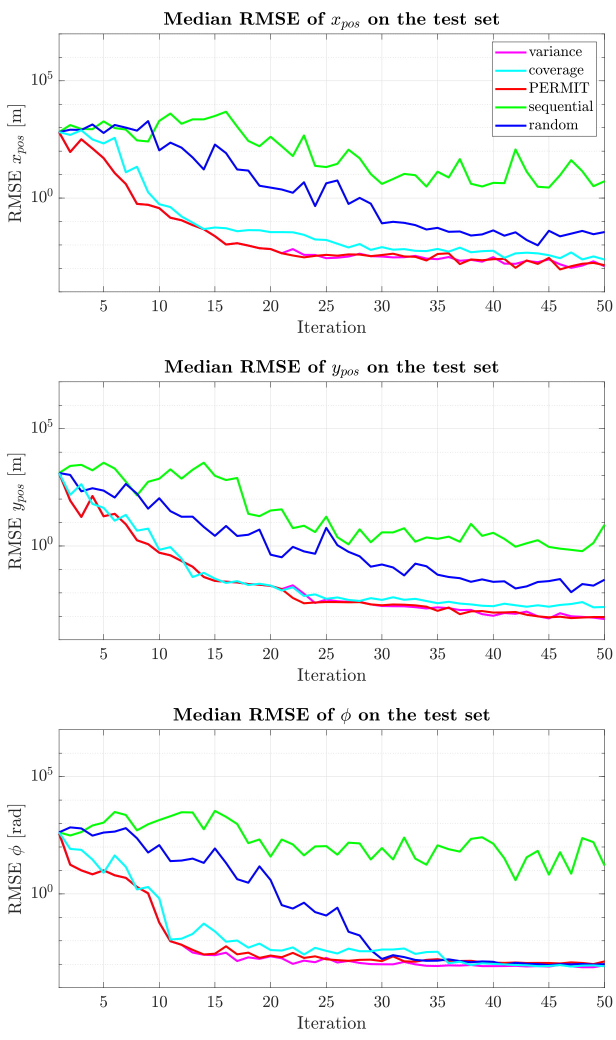


FIGURE 2. Evaluation of the sample-selection methods for modeling all variables in the experiment with the simulated mobile robot. The number of training samples starts at five in the first iteration and increases by one in each iteration.

The results in Table 1 and in Fig. 2 show that the informative sample-selection methods allow for constructing accurate

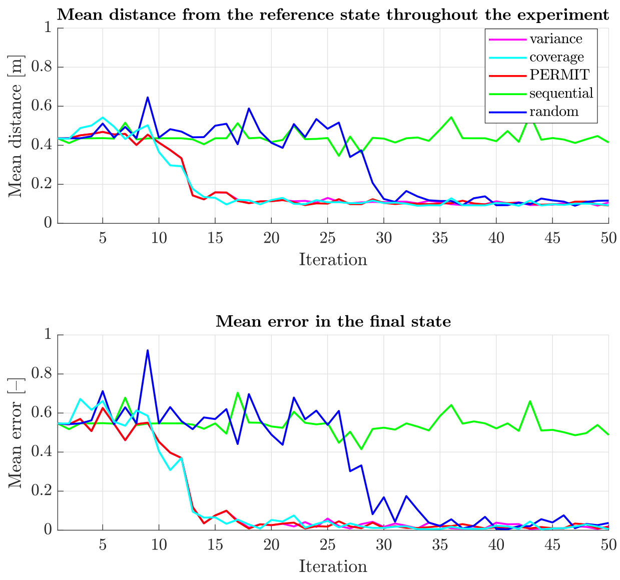


FIGURE 3. Evaluation of the control task executions performed with the simulated mobile robot.

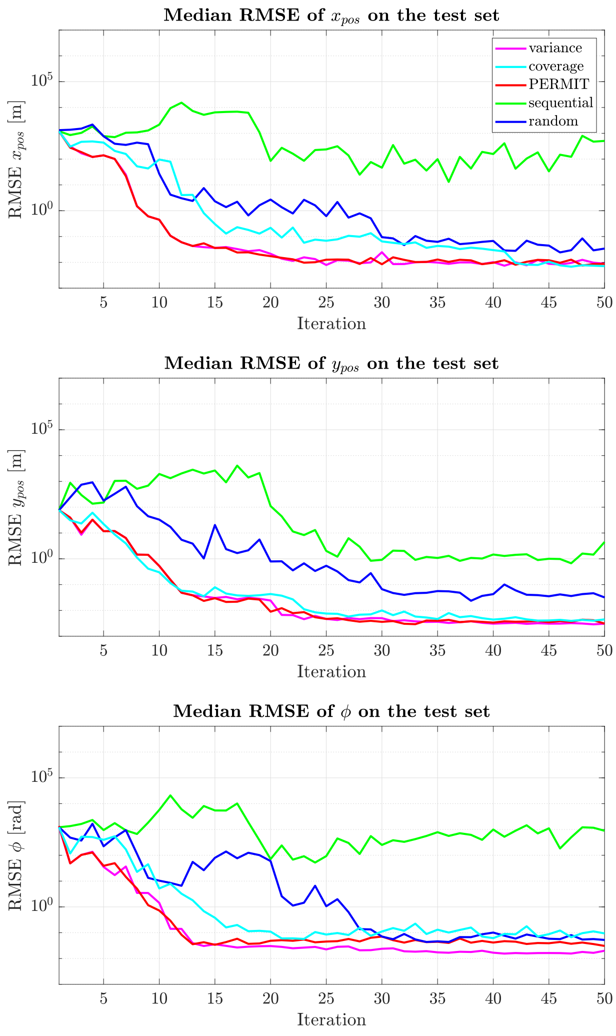


FIGURE 4. Evaluation of the sample-selection methods for modeling all variables in the experiment with the real mobile robot. The number of training samples starts at five in the first iteration and increases by one in each iteration.

analytic models using substantially fewer training samples as compared to the sequential and random approach. While the PERMIT method and the variance method have similar overall performance, they both slightly outperform the maximum output domain coverage method. Note that all methods start with the initial training set composed of only the first five samples in the buffer, resulting in highly overfitted models yielding large errors on the test set. Therefore, the mean values in Table 1 may appear relatively large, which is induced by the large errors in the first iterations. However, the generalization ability of the models constructed using the informed sample-selection methods rapidly improves with the increasing size of the training set, as illustrated in Fig. 2. For instance, the median RMSE for x_{pos} drops to approx. 0.1 m with 16 training samples (12th iteration) and further to approx. 0.01 m with 20 training samples (16th iteration) for the PERMIT method and for the variance method.

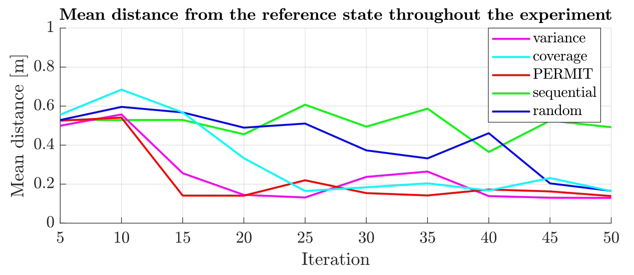
The results of the control task simulations are presented in Fig. 3. After a few initial iterations, the informed sample-selection methods clearly outperform the sequential and random method. The random method reaches an acceptable performance around the 30th iteration, but its performance in the following iterations oscillates. In contrast, the informed methods steadily construct well-performing models. Note that the performance of the sequential approach is very poor. This is because the first 54 samples in the buffer do not contain information that would help to improve the accuracy of the model. Such a situation is encountered in many real scenarios.

F. RESULTS WITH THE REAL MOBILE ROBOT

We have performed lab experiments with TurtleBot 2, see Fig. 1b. The robot has collected the data samples as described in Section III-B, yielding 500 samples in the buffer and 250 samples in the test set.

The quality of the models measured by RMSE on the test data set throughout the execution of the model learning algorithm is shown in Fig. 4 and the quantitative results are summarized in Table 1. Similar conclusions as for the simulated mobile robot can be drawn. The PERMIT method and the variance method perform the best, followed by the coverage method. The informed sample-selection methods substantially outperform the sequential and random method.

Measures of the control task performance are shown in Fig. 5. On the control task with the real mobile robot, the PERMIT method is among the fastest ones to achieve a good performance. The variance method performs also very well, followed by the coverage method. The random method only achieves an acceptable performance at the end of the experiment, using 54 training samples. The sequential method performs the worst.



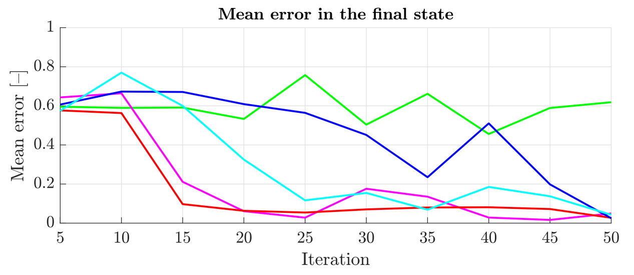


FIGURE 5. Evaluation of the control task executions performed with the real mobile robot.

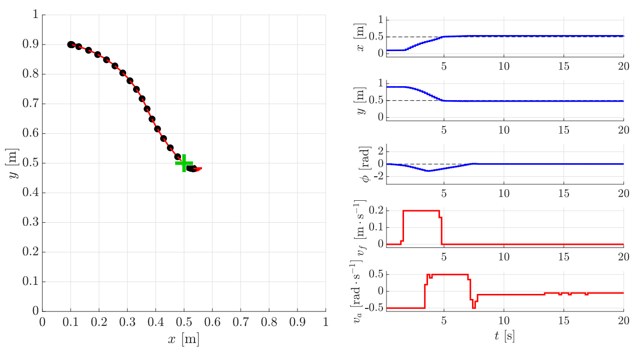


FIGURE 6. Execution of the control task performed on the real robot with an analytic model trained on a data set of only 24 data samples, selected by the PERMIT method. The plot shows the trajectory of the robot (solid black circles) with its orientation (red markers). The reference state is marked by a green cross.

An example of the control task execution on the real robot is shown in Fig. 6. The RL controller is based on an analytic model trained on only 24 data samples, which were selected by the PERMIT method. The results show that the model constructed by the proposed method allows to build an RL-based controller that performs the control task well. The lab experiment is captured in the video attachment, also available at our GitHub repository.[1]

IV. DRONE EXPERIMENTS

We have selected the Parrot Bebop 2 drone to demonstrate the performance of the method on higher-dimensional problems.

A. SYSTEM DESCRIPTION

The output variables are the translational velocities v_x, v_y, and v_z (measured by the OptiTrack motion-capture system in the fixed world frame) and the body angles θ, φ and ψ, denoting the pitch, roll, and yaw, respectively. The drone is controlled by θ_c, φ_c, ω_c, v_{z_c}, which denote the desired roll, pitch, yaw rate, and vertical velocity, respectively. Fig. 7 shows a schematic and a photo of the drone.

[1]https://github.com/erik-derner/sample-selection/blob/master/TurtleBot_ModelLearning.mp4

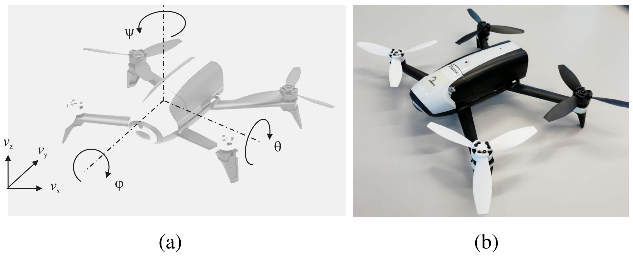


FIGURE 7. Parrot Bebop Drone: a) schematic, b) photo of the quadcopter used in the experiments.

In the experiments, we use the sampling period $T_s = 0.05$ s. We have chosen a smaller sampling period than for the mobile robot in order to capture the faster movement of the drone.

B. DATA COLLECTION

We have collected 1722 data samples by teleoperating the drone to follow a given trajectory. This data set was divided into a training set and a test set in the ratio 2:1 by moving every third sample to the test set.

The models, constructed for each state variable, are in the input–output form $\mathbf{g}(\mathbf{y}_k, \mathbf{y}_{k-1}, \ldots, \mathbf{u}_k, \mathbf{u}_{k-1}, \ldots)$, where $\mathbf{y}$ denotes the vector of output variables and $\mathbf{u}$ are the control inputs. We use a first-order model for the translational velocities v_x, v_y and a second-order model for all the other variables.

C. MODEL LEARNING

The model learning algorithm starts with a training data set of $n_0 = 5$ first samples in the buffer. As there are 1148 samples in the buffer, the aim is to select a small subset of data that will capture the robot's dynamics as well as possible. In each iteration, $n_r = 10$ analytic models are constructed for each variable with the same SNGP configuration as for the mobile robot. Only $n_s = 1$ sample is added in each iteration to the training data set for each variable. The number of iterations is limited to $n_i = 25$.

D. RESULTS

The performance of the five sample-selection methods of Section II-C is shown in Fig. 8. The quantitative measures are summarized in Table 2. All angles and their differences have been wrapped to the domain $(-\pi, \pi]$ rad.

The results show that for modeling v_x, v_y, and v_z, the PERMIT method and the variance method achieve the best performance. The coverage method and the random method are slightly worse. The difference between the first two and the latter two methods increases for the variables θ, φ, and ψ. The sequential method performs the worst for all variables, which

TABLE 2. Comparison of performance of the sample-selection methods on All variables in the experiment with the drone. The RMSE median and the RMSE spread Are averaged over All iterations of the sample-selection procedure. The RMSE spread Is calculated as the difference between the maximum and minimum error among the models in each iteration.

	Selection method	$v_x\ [\mathrm{m}\cdot\mathrm{s}^{-1}]$	$v_y\ [\mathrm{m}\cdot\mathrm{s}^{-1}]$	$v_z\ [\mathrm{m}\cdot\mathrm{s}^{-1}]$	θ [rad]	φ [rad]	ψ [rad]
Average RMSE median	variance	$2.41\cdot10^{1}$	$1.06\cdot10^{1}$	$8.22\cdot10^{1}$	$4.48\cdot10^{-1}$	$3.39\cdot10^{-1}$	$5.91\cdot10^{-1}$
	coverage	$1.60\cdot10^{1}$	$3.29\cdot10^{1}$	$8.07\cdot10^{1}$	$5.51\cdot10^{-1}$	$3.90\cdot10^{-1}$	$6.91\cdot10^{-1}$
	PERMIT	$7.03\cdot10^{0}$	$1.16\cdot10^{1}$	$8.61\cdot10^{1}$	$3.87\cdot10^{-1}$	$2.99\cdot10^{-1}$	$5.86\cdot10^{-1}$
	sequential	$3.42\cdot10^{6}$	$2.83\cdot10^{4}$	$1.11\cdot10^{3}$	$1.76\cdot10^{0}$	$1.64\cdot10^{0}$	$1.81\cdot10^{0}$
	random	$1.44\cdot10^{1}$	$2.21\cdot10^{1}$	$7.80\cdot10^{1}$	$4.84\cdot10^{-1}$	$4.18\cdot10^{-1}$	$6.45\cdot10^{-1}$
Average RMSE spread	variance	$7.04\cdot10^{6}$	$3.64\cdot10^{4}$	$2.93\cdot10^{5}$	$6.60\cdot10^{-1}$	$5.74\cdot10^{-1}$	$4.82\cdot10^{-1}$
	coverage	$3.25\cdot10^{4}$	$1.02\cdot10^{7}$	$1.68\cdot10^{18}$	$8.67\cdot10^{-1}$	$8.20\cdot10^{-1}$	$7.53\cdot10^{-1}$
	PERMIT	$5.24\cdot10^{3}$	$3.56\cdot10^{4}$	$2.93\cdot10^{5}$	$6.14\cdot10^{-1}$	$6.27\cdot10^{-1}$	$5.95\cdot10^{-1}$
	sequential	$1.28\cdot10^{65}$	$4.55\cdot10^{22}$	$4.27\cdot10^{27}$	$9.43\cdot10^{-1}$	$1.31\cdot10^{0}$	$6.03\cdot10^{-1}$
	random	$2.39\cdot10^{6}$	$7.56\cdot10^{4}$	$2.93\cdot10^{5}$	$7.53\cdot10^{-1}$	$9.26\cdot10^{-1}$	$7.66\cdot10^{-1}$

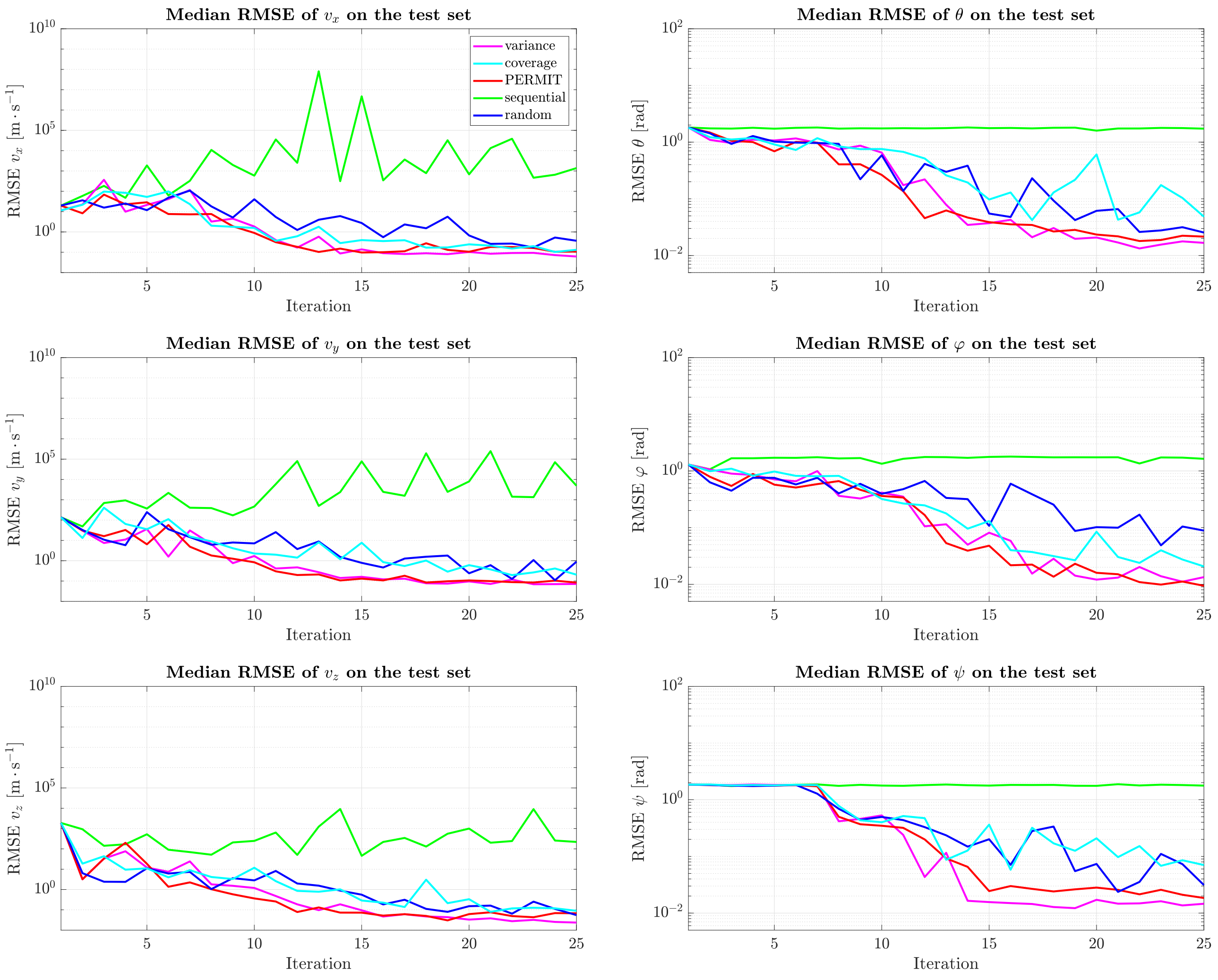


FIGURE 8. Evaluation of the sample-selection methods for modeling all variables in the experiment with the drone. The number of training samples starts at five in the first iteration and increases by one in each iteration.

is due to the absence of a sufficient number of informative samples at the beginning of the recorded sequence. On the other hand, the whole buffer contains a larger amount of informative samples than in the case of the mobile robot, which makes the random method perform better compared to the results in Section III. Overall, the results show that using informed sample selection allows SR to find accurate models from small batches of data (20–25 samples) also on a higher-dimensional problem.

V. CONCLUSION

The selection of training samples is essential to efficiently construct accurate nonlinear dynamic models from the vast amount of collected data. Not all data samples are equally informative: some carry unique information about the system, while others are redundant. To that end, we have proposed an approach for constructing compact training data sets that serve as an input to a model learning method. For model learning, we have chosen symbolic regression thanks to its ability to construct accurate models in the form of analytic equations even from small data sets.

Sample-selection methods can be classified into uninformed and informed methods. As a baseline, we have included two uninformed methods that select the training samples sequentially and randomly. Informed methods in contrast select the training samples based on predefined criteria with the aim to capture the important properties of the system and so to achieve a better modeling accuracy. We have evaluated two state-of-the-art informed sample-selection methods, based on the model variance and on the output domain coverage. In addition, we have proposed a novel sample-selection method based on the model prediction error, called PERMIT.

We have evaluated the methods on data from three dynamic systems: a simulated mobile robot, a real mobile robot TurtleBot 2, and a real Parrot Bebop drone. All three informed sample-selection techniques clearly outperform the two baseline uninformed methods: they quickly select a small subset of important samples from a large data buffer. While PERMIT and the variance method achieve the best performance, the results of the coverage method are slightly worse in the overall evaluation. For the PERMIT method, we have shown that an analytic model found by symbolic regression on a training data set with as few as 24 samples can already be used to design a near-optimal RL controller for the real mobile robot.

In our future work, we will conduct a real-world, long-term autonomy experiment to evaluate how the sample-selection methods perform in a setting where unexpected events can occur, including data loss, sensor faults, etc. Even though the proposed method is robust to a small number of outliers in the training data set, the accuracy of the models will be affected if the amount of erroneous data is too large. To address this, we will also investigate methods for automated data set maintenance, including removal of data samples that diminish the accuracy of the models.

ACKNOWLEDGMENT

The authors thank Hai Zhu for collecting the data in the drone experiments.

REFERENCES

[1] L. Ljung, *System Identification: Theory for the User*, 2nd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 1999.

[2] C. P. Neuman and P. K. Khosla, *Identification of Robot Dynamics: An Application of Recursive Estimation*. Boston, MA, USA: Springer, 1986, pp. 175–194.

[3] B. Settles, "Active learning literature survey," Dept. Comput. Sci., Univ. Wisconsin-Madison, Madison, WI, USA, Tech. Rep. 1648, 2009.

[4] T. de Bruin, J. Kober, K. Tuyls, and R. Babuska, "Integrating state representation learning into deep reinforcement learning," *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, pp. 1394–1401, Jul. 2018.

[5] J. Bongard and H. Lipson, "Automated reverse engineering of nonlinear dynamical systems," *Proc. Nat. Acad. Sci. USA*, vol. 104, no. 24, pp. 9943–9948, Jun. 2007.

[6] H. O. Ilhan and M. F. Amasyal, "Comparing informative sample selection strategies in classification ensembles," *Int. J. Mach. Learn. Comput.*, vol. 4, no. 1, p. 79, 2014.

[7] J. Yao, Y. Wu, J. Koo, B. Yan, and H. Zhai, "Active learning algorithm for computational physics," 2019, *arXiv:1904.10692*. [Online]. Available: http://arxiv.org/abs/1904.10692

[8] S. Ertekin, J. Huang, L. Bottou, and L. Giles, "Learning on the border: Active learning in imbalanced data classification," in *Proc. 16th ACM Conf. Inf. Knowl. Manage.*, 2007, pp. 127–136.

[9] N. Khoshnevis and R. Taborda, "Application of pool-based active learning in physics-based earthquake ground-motion simulation," *Seismol. Res. Lett.*, vol. 90, no. 2A, pp. 614–622, 2019.

[10] D. Wu, C.-T. Lin, and J. Huang, "Active learning for regression using greedy sampling," *Inf. Sci.*, vol. 474, pp. 90–105, Feb. 2019.

[11] R. Burbidge, J. J. Rowland, and R. D. King, "Active learning for regression based on query by committee," in *Proc. Int. Conf. Intell. Data Eng. Autom. Learn.* Berlin, Germany: Springer, 2007, pp. 209–218.

[12] E. Derner, J. Kubalík, N. Ancona, and R. Babuška, "Constructing parsimonious analytic models for dynamic systems via symbolic regression," *Appl. Soft Comput.*, vol. 94, Sep. 2020, Art. no. 106432.

[13] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[14] N. Heess, G. Wayne, D. Silver, T. Lillicrap, T. Erez, and Y. Tassa, "Learning continuous control policies by stochastic value gradients," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 2944–2952.

[15] A. Punjani and P. Abbeel, "Deep learning helicopter dynamics models," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May 2015, pp. 3223–3230.

[16] J. Kubalík, E. Derner, and R. Babuška, "Enhanced symbolic regression through local variable transformations," in *Proc. 9th Int. Joint Conf. Comput. Intell.* Lisboa, Portugal: SciTePress, Nov. 2017, pp. 91–100.

[17] E. Derner, J. Kubalík, and R. Babuška, "Reinforcement learning with symbolic input-output models," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 3004–3009.

[18] M. P. Deisenroth, *Efficient Reinforcement Learning Using Gaussian Processes*, vol. 9. Karlsruhe, Germany: KIT Scientific Publishing, 2010.

[19] L. Buşoniu, D. Ernst, B. De Schutter, and R. Babuška, "Approximate dynamic programming with a fuzzy parameterization," *Automatica*, vol. 46, no. 5, pp. 804–814, May 2010.

[20] J. C. Butcher and N. Goodwin, *Numerical Methods for Ordinary Differential Equations*, vol. 2. Hoboken, NJ, USA: Wiley, 2008.

ERIK DERNER (Student Member, IEEE) received the M.Sc. degree (Hons.) in artificial intelligence and computer vision from Czech Technical University (CTU) in Prague, where he is currently pursuing the Ph.D. degree. His research interests include long-term autonomy, mobile robotics, genetic programming, reinforcement learning, and computer vision. He currently focuses on life-long autonomy in robot control and navigation. The central topic in his research is the use of symbolic regression to automatically construct nonlinear models of dynamic systems.

JIŘÍ KUBALÍK received the M.Sc. degree in computer science and the Ph.D. degree in artificial intelligence and biocybernetics from the Czech Technical University in Prague, in 1994 and 2001, respectively. He is currently a Researcher with the Czech Institute of Informatics, Robotics, and Cybernetics, CTU in Prague. His research has mainly been focused on various types of evolutionary computation techniques and their applications to hard optimization problems. He currently focuses on developing symbolic regression methods to construct analytic models of dynamic systems with the use of prior knowledge about the systems.

ROBERT BABUŠKA, (Member, IEEE) received the M.Sc. degree (Hons.) in control engineering from the Czech Technical University in Prague, in 1990, and the Ph.D. degree *(cum laude)* from the Delft University of Technology, The Netherlands, in 1997. He has had faculty appointments with the Czech Technical University in Prague and the Faculty of Electrical Engineering, TU Delft. He is currently a Full Professor of intelligent control and robotics with the Department of Cognitive Robotics, Faculty of Mechanical, Maritime and Materials Engineering, TU Delft. In the past, he made seminal contributions to the field of nonlinear control and identification with the use of fuzzy modeling techniques. His current research interests include reinforcement learning, adaptive and learning robot control, nonlinear system identification, and state estimation. He has been involved in the applications of these techniques in various fields, ranging from process control to robotics and aerospace.