Guiding Robot Model Construction with Prior Features

Erik Derner[1], Jiří Kubalík[2], and Robert Babuška[3]

***Abstract*— Virtually all robot control methods benefit from the availability of an accurate mathematical model of the robot. However, obtaining a sufficient amount of informative data for constructing dynamic models can be difficult, especially when the models are to be learned during robot deployment. Under such circumstances, standard data-driven model learning techniques often yield models that do not comply with the physics of the robot. We extend a symbolic regression algorithm based on Single Node Genetic Programming by including the prior model information into the model construction process. In this way, symbolic regression automatically builds models that compensate for theoretical or empirical model deficiencies. We experimentally demonstrate the approach on two real-world systems: the TurtleBot 2 mobile robot and the Parrot Bebop 2 drone. The results show that the proposed model-learning algorithm produces realistic models that fit well the training data even when using small training sets. Passing the prior model information to the algorithm significantly improves the model accuracy while speeding up the search.**

***Index Terms*— Model learning for control, AI-based methods, genetic programming, prior knowledge.**

I. Introduction

To guarantee long-term robot autonomy, methods are needed to automatically build and update dynamic models of robots and their environments. Techniques for learning models from data samples collected during routine robot operation inevitably have to deal with imperfections of the measured data, such as uneven sample distribution, limited sensor accuracy, presence of noise, etc. However, some partial information about the robot model is often known, such as a theoretical or empirical model.

A range of techniques can be employed to learn a non-linear model of the robot dynamics, as presented in the surveys [1], [2], [3]. Many of the popular methods such as deep neural networks [4] and locally weighted learning [5] are black-box, require a large amount of high-quality training data, and do not allow for the use of prior knowledge in the form of partial models or constraints. In general, obtaining appropriate training sets is difficult and often not even possible [6], in particular during the robot deployment. To this end, symbolic regression (SR) allows to naturally incorporate the prior knowledge and it is able to learn from very small training sets. SR evolves models in the form of analytic expressions by combining user-defined elementary functions. Although SR has not yet been widely adopted by the robotics community, its potential has already been demonstrated [7], [8], [9].

In this paper, we extend our previous work [9], [10] by including building blocks in the form of physical or empirical models (or their parts) into the model construction process. The proposed method automatically finds accurate and physically valid robot models by complementing training data with information capturing the desired properties of the model sought. This paper presents the following three main contributions:

- We adapt a SR method based on Single Node Genetic Programming (SNGP) by including a prior model to efficiently construct accurate parsimonious analytic models from small training data sets supported by the prior information.
- The benefits of employing empirical or theoretical models are evaluated against two other SR variants – baseline SR that does not use prior knowledge in any form [9], and SR using formal constraints [10].
- We illustrate on two examples of real robots that the method efficiently compensates for deficiencies in real data and yields models that are both accurate and physically plausible.

The remainder of the paper is organized as follows: Section II gives an overview of the related work. The model construction method is described in Section III. Section IV presents the results of the experimental evaluation and Section V concludes the paper.

II. Related Work

In its basic form, SR allows to incorporate prior knowledge by selecting the set of elementary functions that are used in the inner nodes of the tree-based model representations [11]. In grammar-based approaches, prior knowledge can be included into the grammar describing the set of available structure elements from which the models are built. An example of a grammar-based approach is the tree adjoining grammar GP [12]. Some SR approaches take into account prior knowledge in the form of formal constraints [10], [13]. Another recent SR approach named AI Feynman [14] exploits known properties of the function sought, such as

This work was supported by the European Regional Development Fund under the project Robotics for Industry 4.0 (reg. no. CZ.02.1.01/0.0/0.0/15_003/0000470) and by the Grant Agency of the Czech Technical University in Prague, grant no. SGS19/174/OHK3/3T/13.

[1]Erik Derner is with the Czech Institute of Informatics, Robotics and Cybernetics, Czech Technical University in Prague, 16000 Prague, Czech Republic and with the Department of Control Engineering, Faculty of Electrical Engineering, Czech Technical University in Prague, 12135 Prague, Czech Republic erik.derner@cvut.cz

[2]Jiří Kubalík is with the Czech Institute of Informatics, Robotics and Cybernetics, Czech Technical University in Prague, 16000 Prague, Czech Republic jiri.kubalik@cvut.cz

[3]Robert Babuška is with Cognitive Robotics, Faculty of 3mE, Delft University of Technology, 2628 CD Delft, The Netherlands and with the Czech Institute of Informatics, Robotics and Cybernetics, Czech Technical University in Prague, 16000 Prague, Czech Republic r.babuska@tudelft.nl

physical units for dimensional analysis, or symmetry with respect to some of its variables.

Methods for incorporating prior knowledge also exist for neural networks, such as [15], inspired by Hamiltonian mechanics. A neural network is trained to learn and follow the basic laws of physics. Gaussian processes [16], [17] and Koopman operators [18], [19], [20] also allow for incorporating prior knowledge to the model construction process. Gaussian processes are non-parametric and the Koopman operator methods require in theory an infinitely large set of bases, in practice orders of magnitude larger than the number of regressors. In contrast to these methods, we aim at constructing parsimonious non-linear models.

The literature on including partial empirical or theoretical models in data-driven robot model construction is limited. It has been addressed mainly in the context of local modeling [21], [22]. However, these techniques require a human expert to define validity regions for the individual submodels and as such are unsuitable for automated procedures required by the application to long-term robot autonomy.

In our previous work [10], we have shown that the model construction process can be guided towards a physically meaningful model through formal constraints. In this work, we suggest another approach to incorporate known robot properties in the form of a partial model. In most cases, the theoretical or empirical model of the robot is known in advance and it can be naturally plugged into the SR-driven model construction process as a building block which allows to find more accurate models faster. The apriori model information can be used both in standard SR [9] as well as in SR with formal constraints [10].

III. Method

In this section, we first present a brief overview of the baseline SR and then we extend it to include prior knowledge.

A. Baseline Symbolic Regression

Symbolic regression uses genetic programming to build from data non-linear models in the following form:

$$y = f(\xi), \tag{1}$$

where $\xi \in \mathcal{X} \subset \mathbb{R}^n$ is the model input and $y \in \mathcal{Y} \subset \mathbb{R}$ is its output. The method can be applied to both discrete-time and continuous-time dynamic models. In this paper, we consider state-space models in continuous time

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \tag{2}$$

with the state $\mathbf{x} \in \mathcal{S} \subset \mathbb{R}^s$, the control input $\mathbf{u} \in \mathcal{U} \subset \mathbb{R}^m$, and the state derivative $\dot{\mathbf{x}}$. Continuous-time models allow to naturally incorporate prior knowledge based on physics, which will be demonstrated in Section IV. Note that we use SR to model the individual state derivative components independently. In the sequel, a model for a generic state derivative component $\dot{x}$ will be denoted as $\hat{\dot{x}} = f(\mathbf{x}, \mathbf{u})$. In some cases, the state derivatives can be directly measured, while in other cases they need to be approximated using a difference operator on the discrete-time state measurements.

We use an adapted version of the SNGP algorithm, originally described in [11]. SNGP stores functions represented as tree structures in a single linear array, where each element of the array corresponds to one node. The array of nodes represents a population. A node may be either an elementary function, such as addition or multiplication, or a terminal, i.e., a variable or a constant. Nodes representing functions may only take as their arguments nodes that appear earlier in the array.

Instead of evolving the model as a single function, we adopt the approach presented in [23] and construct the model as a composition of genetically evolved non-linear functions f_i, called *features*:

$$f(\xi) = \beta_0 + \sum_{i=1}^{n_f} \beta_i f_i(\xi), \tag{3}$$

where the coefficients β_i, $i = 0, 1, \ldots, n_f$ are estimated by least squares. The features f_i represent mathematical expressions composed of elementary functions and terminals. They are evolved through an iterative process using a mutation operator. The main user-defined parameters include the elementary function set $\mathcal{F}$, the population size n_i, the number of generations n_g, the maximum number of features n_f, and the maximum depth d of the trees representing the features. The evolution is driven by a fitness function minimizing the root-mean-square error (RMSE) on the training data set, denoted e_d^{train}. For more details on the SNGP implementation used in this work, please refer to [24].

B. Prior Knowledge

Prior knowledge of the model's properties can be included in the model construction process as a partial model or as formal constraints. First, we describe how to represent prior knowledge in the form of a partial model.

An approximate or partial theoretical or empirical model of the robot is often known. This information can be included in the model structure as prior features:

$$f(\xi) = \beta_0 + \underbrace{\sum_{i=1}^{n_p} \beta_i \bar{f}_i(\xi)}_{\text{prior features}} + \underbrace{\sum_{i=n_p+1}^{n_f} \beta_i f_i(\xi)}_{\text{evolved features}}. \tag{4}$$

Features $\bar{f}_i(\xi)$ encode the prior knowledge in the form of fixed functions specified by the user in advance, while features $f_i(\xi)$ are evolved by genetic programming to compensate for the prior features' deficiency manifested by an error in fitting the training data. All the coefficients β_i, $i \in \{0, \ldots, n_f\}$, are estimated using least squares, i.e., both for $\bar{f}_i(\xi)$ and $f_i(\xi)$. In many cases, the apriori model information will be stored in only one prior feature $\bar{f}_1$, $n_p = 1$. However, the above formulation allows for decomposing the theoretical or empirical model into several features. In this way, some of its inner parameters can also be tuned.

Prior knowledge can also be given through formal constraints. Desired model properties, such as monotonicity

or symmetry, can be written as equality and inequality constraints:

$$c_i^E(\xi) = 0 \quad \forall\, \xi \in \mathcal{C}_i^E \subset \mathcal{X} \tag{5}$$

with $i \in \{1,\ldots,n_c^E\}$, and

$$c_i^I(\xi) \leq 0 \quad \forall\, \xi \in \mathcal{C}_i^I \subset \mathcal{X} \tag{6}$$

with $i \in \{1,\ldots,n_c^I\}$. For each constraint, a given number of samples is randomly drawn from a uniform distribution over the specified constraint domain $\mathcal{C}_i^E$ and $\mathcal{C}_i^I$. This yields two sets of samples: D_i^E containing samples for each equality constraint $c_i^E(\xi)$ and D_i^I with samples for each inequality constraint $c_i^I(\xi)$. The *constraint violation error*, denoted e_c, takes into account both types of constraints. We refer the interested reader to [10] for details.

The above formal constraints are incorporated in the model search in the form of multi-objective optimization. The bi-objective SNGP simultaneously optimizes the model with respect to a) the RMSE on the training data set e_d^{train}, and b) the RMSE on the training constraint set e_c^{train}.

Among the generated models, the user can select the desired trade-off between the accuracy of fitting the data and the constraint satisfaction. In this work, we determine the best model as the one that minimizes the training data error e_d^{train} among all the models that have the constraint satisfaction error e_c^{train} below a given threshold γ.

IV. Experiments

We have selected two robotic benchmarks to evaluate our method: a mobile robot TurtleBot 2 and a drone Parrot Bebop 2. These two robots were chosen to assess the performance of the proposed method on systems with different dynamics, complexity, and types of nonlinearities.

A. Evaluation Scheme

We consider two scenarios: baseline SNGP and SNGP with formal constraints. The former fits the data minimizing the error e_d^{train}, while the latter performs a multi-objective optimization including formal constraints and also minimizing the error e_c^{train}.

In both scenarios, we compare a variant of SNGP without prior features and including prior features. This allows to measure the benefit of including prior features, while the total number of features as well as other SNGP parameters remain the same, see Section IV-B.

We use the following data sets for training and evaluation:

1) Training and Test Data Sets: The training and test data sets are two disjoint sets collected during the robot operation. The training set is used only for learning the model, while the test set serves to measure the error e_d^{test}, capturing how well the model fits previously unseen data.

2) Training and Test Constraint Sets: A set of training constraint samples is drawn from a uniform distribution for each of the defined constraints to evaluate the constraint violation error e_c^{train} in the multi-objective model search. The test constraint set is drawn from the same distribution to measure how well the model satisfies the required formal constraints on data different from the training samples.

TABLE I

Main SNGP parameters used in the experiments.

Parameter	Symbol	Value
Elementary function set	$\mathcal{F}$	{+, −, ×, Square, Cube, Sine, Cosine}
Population size	n_i	500
Number of generations	n_g	60 000
Maximum number of features	n_f	10
Maximum tree depth	d	7

We report the median errors e_d^{test} and e_c^{test} on the test data set and on the test constraint set, respectively. The median errors are calculated over 50 runs of SNGP.

A comparison of the SNGP performance with alternative modeling methods such as neural networks or local linear regression has been presented in our previous work [9], [25] and is beyond the scope of this paper.

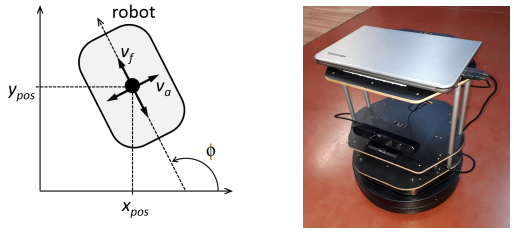
B. Method Parameters and Complexity

We use the default configuration for all experiments. The main parameters of all variants of SNGP evaluated in the experiments are summarized in Table I. Note that the prior features also count towards the maximum number of features n_f to ensure a fair comparison.

We have empirically observed across various data sets that the computational complexity of symbolic regression grows linearly with the number of samples and with the number of generations. A single run of SNGP takes 2–7 minutes on a standard PC[1] for the experiments described in this section, depending on the training data set size.

C. Mobile Robot

We have recorded the training and test data sets on a real robot TurtleBot 2, see Fig. 1.



(a) (b)

Fig. 1. Mobile robot TurtleBot 2: a) schematic, b) photograph.

1) System Description: The state of the two-wheeled mobile robot is defined as $\mathbf{x} = (x_{pos}, y_{pos}, \phi)^\top$, 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 robot is:

$$\dot{x}_{pos} = v_f \cos\phi\,, \tag{7}$$

$$\dot{y}_{pos} = v_f \sin\phi\,, \tag{8}$$

$$\dot{\phi} = v_a\,. \tag{9}$$

[1]CPU Intel Core i7-4610M @ 3.00 GHz, 16 GB RAM

We are interested in finding the continuous-time model of $\dot{x}_{pos}$ and $\dot{y}_{pos}$ fitting the measured data:

$$\hat{\dot{x}}_{pos} = f_{\dot{x}_{pos}}(x_{pos}, y_{pos}, \phi, v_f, v_a), \tag{10}$$

$$\hat{\dot{y}}_{pos} = f_{\dot{y}_{pos}}(x_{pos}, y_{pos}, \phi, v_f, v_a). \tag{11}$$

Modeling of $\dot{\phi}$ is omitted from this paper for its simplicity. We use a data set of 130 samples both for $\dot{x}_{pos}$ and $\dot{y}_{pos}$. The data set was divided in the ratio 2:1 into 87 training samples and 43 test samples. The discrete-time data samples were recorded with a sampling period $T_s = 0.2\,\mathrm{s}$ while the robot was moving along a random trajectory. Given the samples composed of the current state $\mathbf{x}_k$, current input $\mathbf{u}_k$, and the next state $\mathbf{x}_{k+1}$, we used the forward difference to approximate the state derivatives.

Note that in (10) and (11), all the state and input variables are available to SR, even though they are not needed according to the theoretical models (7), (8). SR automatically chooses the variables that are useful to capture the properties of the function fitting the data.

2) Prior Knowledge: The theoretical models (7) and (8) are used as the prior features $\bar{f}_1$, $\bar{g}_1$ in learning the models $\hat{\dot{x}}_{pos}$ and $\hat{\dot{y}}_{pos}$, respectively. Symbolic regression allows for modeling imperfections that are not captured in the theoretical models, such as friction, and it can also compensate for sensor inaccuracies.

We define three formal constraints for modeling $\dot{x}_{pos}$:

1) The robot must have a zero velocity along the x-axis if the linear velocity v_f is zero: $x_{pos} \in [-10, 10]\,\mathrm{m}$, $y_{pos} \in [-10, 10]\,\mathrm{m}$, $\phi \in (-\pi, \pi]\,\mathrm{rad}$, $v_f = 0\,\mathrm{m \cdot s^{-1}}$, $v_a \in [-\pi, \pi]\,\mathrm{rad \cdot s^{-1}} \rightarrow \dot{x}_{pos} = 0\,\mathrm{m \cdot s^{-1}}$.
2) The robot must have a zero velocity along the x-axis if it is moving in the positive direction of the y-axis and not rotating at the same time: $x_{pos} \in [-10, 10]\,\mathrm{m}$, $y_{pos} \in [-10, 10]\,\mathrm{m}$, $\phi = \pi/2\,\mathrm{rad}$, $v_f \in [-0.6, 0.6]\,\mathrm{m \cdot s^{-1}}$, $v_a = 0\,\mathrm{rad \cdot s^{-1}} \rightarrow \dot{x}_{pos} = 0\,\mathrm{m \cdot s^{-1}}$.
3) The robot must have a zero velocity along the x-axis if it is moving in the negative direction of the y-axis and not rotating at the same time: $x_{pos} \in [-10, 10]\,\mathrm{m}$, $y_{pos} \in [-10, 10]\,\mathrm{m}$, $\phi = -\pi/2\,\mathrm{rad}$, $v_f \in [-0.6, 0.6]\,\mathrm{m \cdot s^{-1}}$, $v_a = 0\,\mathrm{rad \cdot s^{-1}} \rightarrow \dot{x}_{pos} = 0\,\mathrm{m \cdot s^{-1}}$.

Similarly, we define three additional constraints for modeling $\dot{y}_{pos}$, with the only difference in setting $\phi = 0\,\mathrm{rad}$ in the second constraint and $\phi = \pi\,\mathrm{rad}$ in the third constraint. All three additional constraints yield zero $\dot{y}_{pos}$.

For both $\dot{x}_{pos}$ and $\dot{y}_{pos}$, we have generated 50 training constraint samples and 50 test constraint samples for each of the three constraints.

To select the best model among the models generated by SNGP with formal constraints, we consider models with e_c^{train} below the threshold $\gamma = 5 \times 10^{-2}\,\mathrm{m \cdot s^{-1}}$ and we choose the model with the lowest e_d^{train} among them. The value of γ was chosen empirically and it serves to set the trade-off between the two criteria of the multi-objective optimization.

3) Results and Discussion: The results for all cases are summarized in Table II. A comparison of the error on the test data set e_d^{test} for a variant with the prior feature and without it is shown in Fig. 2 for baseline SNGP and in Fig. 3 for SNGP with formal constraints. The error of the prior feature itself is shown as a reference.

TABLE II

MEDIAN ERROR OVER 50 RUNS OF SNGP ON THE TEST DATA SET e_d^{test} AND ON THE TEST CONSTRAINT SET e_c^{test} FOR THE MOBILE ROBOT EXPERIMENT.

	Scenario	Prior feature	Median e_d^{test} $(\mathrm{m \cdot s^{-1}})$	Median e_c^{test} $(\mathrm{m \cdot s^{-1}})$
$\hat{\dot{x}}_{pos}$	Baseline	Not included	5.920×10^{-3}	4.015×10^{1}
		Included	5.562×10^{-3}	3.744×10^{1}
	Constrained	Not included	5.273×10^{-3}	1.589×10^{-2}
		Included	4.973×10^{-3}	1.806×10^{-2}
$\hat{\dot{y}}_{pos}$	Baseline	Not included	6.414×10^{-3}	5.464×10^{0}
		Included	5.455×10^{-3}	1.574×10^{1}
	Constrained	Not included	6.492×10^{-3}	2.457×10^{-2}
		Included	6.010×10^{-3}	2.431×10^{-2}

Using the prior feature yields results superior to the variant without the prior feature in terms of e_d^{test}. This holds for the baseline SNGP as well as for SNGP with formal constraints, both for $\hat{\dot{x}}_{pos}$ and $\hat{\dot{y}}_{pos}$. We used the Wilcoxon rank-sum test [26] to evaluate the statistical significance of the improvement. The improvement is statistically significant with $p = 3.798 \times 10^{-4}$ for the baseline method and with $p = 5.763 \times 10^{-3}$ for SNGP with constraints, both values reported for $\hat{\dot{x}}_{pos}$. Even a more significant improvement is achieved for $\hat{\dot{y}}_{pos}$ with $p = 3.671 \times 10^{-6}$ for the baseline method and with $p = 6.357 \times 10^{-4}$ for constrained SNGP.

SNGP with constraints achieves a better error e_c^{test} than the baseline SNGP for both $\hat{\dot{x}}_{pos}$ and $\hat{\dot{y}}_{pos}$ by several orders of magnitude. The results also demonstrate the capability of the method to learn accurate models from very small data sets, as the models were learned on only 87 training samples.

An example of an analytic model for $\hat{\dot{x}}_{pos}$ and $\hat{\dot{y}}_{pos}$, found using SNGP with formal constraints and with the prior feature included, has been algebraically simplified using Matlab's Symbolic Math Toolbox to the following form:

$$\begin{aligned}
\hat{\dot{x}}_{pos} &= 8.5 \times 10^{-1} v_f \cos(\phi) - 1.2 \times 10^{-2} \sin(\sin(x_{pos})) \\
&\quad + 1.3 \times 10^{-2} \sin(x_{pos})^2 - 7.7 \times 10^{-3} \cos(\phi)^3 \\
&\quad + 3.7 \times 10^{-3} (\phi + v_a) + 2.8 \times 10^{-3} y_{pos} \cos(\phi + v_a) \\
&\quad - 3.2 \times 10^{-3} (v_f + 2)(\sin(\sin(\phi)) - \cos(\phi)^3 \cos(x_{pos})) \\
&\quad - 1.9 \times 10^{-3} (\phi + v_a)^2 + 1.8 \times 10^{-3} \cos(x_{pos})(\phi - 3.1) \\
&\quad + 5.5 \times 10^{-4} y_{pos} - 4.8 \times 10^{-4} v_f - 3.1 \times 10^{-4},
\end{aligned} \tag{12}$$

$$\begin{aligned}
\hat{\dot{y}}_{pos} &= 8.6 \times 10^{-1} v_f \sin(\phi) - 2.3 \times 10^{-1} \cos(1.4 v_f) \\
&\quad - 9.0 \times 10^{-2} v_f \sin(\cos(v_f^2)) - 8.3 \times 10^{-3} \cos(\phi - 2.8 v_f)^4 \\
&\quad + 5.5 \times 10^{-3} (\phi + v_a) + 7.6 \times 10^{-4} x_{pos} - 1.6 \times 10^{-16} y_{pos}^{18} \\
&\quad - 3.0 \times 10^{-3} \sin(x_{pos})^2 - 5.2 \times 10^{-3} \cos(v_a)^9 \\
&\quad - 6.4 \times 10^{-4} (\phi - 2.8 v_f)^2 (y_{pos} - \sin(x_{pos})) \\
&\quad + 6.1 \times 10^{-5} (y_{pos} - \sin(x_{pos}))^3 + 2.4 \times 10^{-1}.
\end{aligned} \tag{13}$$

For both variables, the first term is the prior feature with its coefficient close to one. The coefficient is approximately 15 % smaller than in the prior model, which may

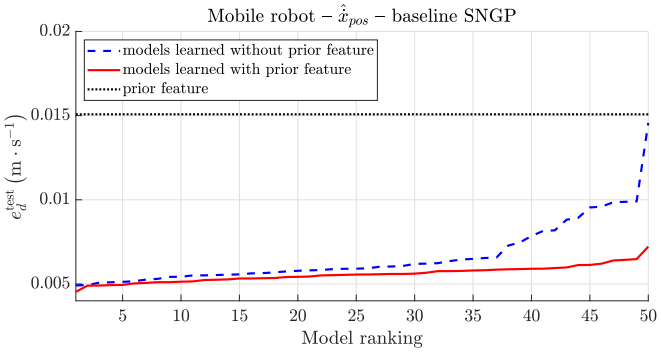


Fig. 2. Comparison of models from 50 baseline SNGP runs in the mobile robot experiment, sorted by RMSE on the test data set e_d^{test}.

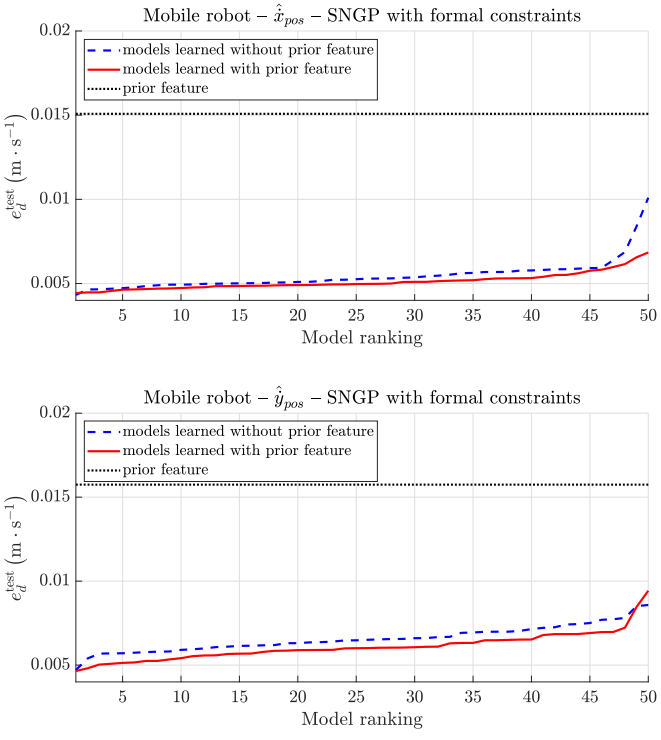
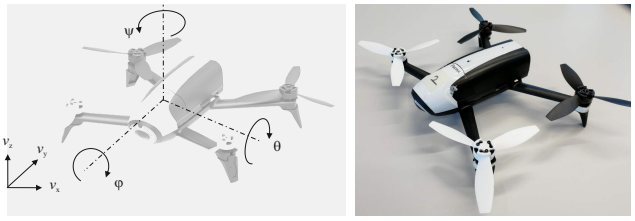


Fig. 3. Comparison of models from 50 runs of SNGP with constraints in the mobile robot experiment, sorted by RMSE on the test data set e_d^{test}.

be explained by the following two reasons. First, the actual forward velocity achieved by the robot is lower than the command velocity v_f. Second, other terms in the model, evolved by combining elementary functions, compensate for inaccuracies of the prior model. This result confirms our hypothesis that the method uses the prior feature as the main building block and constructs the remaining components in the final model to fit the training data.

D. Drone

We have performed experiments with the Parrot Bebop 2 drone, see Fig. 4.



(a) (b)

Fig. 4. Parrot Bebop 2 drone: a) schematic, b) photograph.

1) System Description: The state vector of the drone is $\mathbf{x} = (x, y, z, v_x, v_y, v_z, \theta, \varphi, \psi)^\top$, where x, y, and z denote its position, v_x, v_y, and v_z are the translational velocities measured by the OptiTrack motion-capture system in the fixed world frame, and θ, φ, and ψ are the body angles, denoting the pitch, roll, and yaw, respectively. The drone is controlled by the input vector $\mathbf{u} = (\theta_c,\ \varphi_c,\ \omega_c,\ \omega_{z_c})^\top$, where its components correspond to the desired pitch, roll, and yaw rates and the vertical velocity, respectively. The most complex empirical models are given for $\dot{v}_x$ and $\dot{v}_y$:

$$\dot{v}_x = g\cos\psi\frac{\tan\theta}{\cos\varphi} + g\sin\psi\tan\varphi - k_D v_x\,, \tag{14}$$

$$\dot{v}_y = g\sin\psi\frac{\tan\theta}{\cos\varphi} - g\cos\psi\tan\varphi - k_D v_y\,, \tag{15}$$

where the gravitational acceleration $g = 9.81\,\text{m}\cdot\text{s}^{-2}$ and the drag constant $k_D = 0.28\,\text{s}$ have been estimated empirically. Therefore, we will evaluate our method on modeling these two most challenging variables. We employ SR to build the models of $\dot{v}_x$ and $\dot{v}_y$ from the data in the following form:

$$\hat{\dot{v}}_x = f_{\dot{v}_x}(v_x, \theta, \varphi, \psi)\,, \tag{16}$$

$$\hat{\dot{v}}_y = f_{\dot{v}_y}(v_y, \theta, \varphi, \psi)\,. \tag{17}$$

We have recorded the training data set of 160 samples by steering the real drone to follow an eight-shaped trajectory. The test data set of 251 samples was captured on a square-shaped trajectory. The discrete-time data set for both variables $\dot{v}_x$ and $\dot{v}_y$ was recorded with a sampling period $T_s = 0.05\,\text{s}$. Same as for the mobile robot, we used the forward difference to approximate the derivatives.

2) Prior Knowledge: The theoretical models (14) and (15) are used as the prior features $\bar{f}_1$, $\bar{g}_1$ in learning the models $\hat{\dot{v}}_x$ and $\hat{\dot{v}}_y$, respectively. As the empirical models are composed of three separate terms, alternatively, three prior features can be formulated for $\dot{v}_x$:

$$\begin{aligned} \bar{f}_1 &= g\cos\psi\tan\theta/\cos\varphi, \\ \bar{f}_2 &= g\sin\psi\tan\varphi, \\ \bar{f}_3 &= -k_D v_x \end{aligned} \tag{18}$$

and analogously for $\dot{v}_y$:

$$\begin{aligned} \bar{g}_1 &= g\sin\psi\tan\theta/\cos\varphi, \\ \bar{g}_2 &= -g\cos\psi\tan\varphi, \\ \bar{g}_3 &= -k_D v_y\,. \end{aligned} \tag{19}$$

This enables the method to tune also the coefficients of the empirical model components through least squares, see Section III-B.

We define four formal constraints for modeling $\dot{v}_x$:

1) Given a zero velocity along the x-axis, zero pitch, yaw orienting the drone in the positive direction of the x-axis, and a non-zero roll, the acceleration in the direction of the x-axis has to be zero: $v_x = 0\,\mathrm{m\cdot s^{-1}}$, $\theta = 0\,\mathrm{rad}$, $\varphi \in [-\pi/15, \pi/15]\,\mathrm{rad}$, $\psi = 0\,\mathrm{rad} \rightarrow \dot{v}_x = 0\,\mathrm{m\cdot s^{-2}}$.
2) Given a zero velocity along the x-axis, zero pitch, yaw orienting the drone in the negative direction of the x-axis, and a non-zero roll, the acceleration in the direction of the x-axis has to be zero: $v_x = 0\,\mathrm{m\cdot s^{-1}}$, $\theta = 0\,\mathrm{rad}$, $\varphi \in [-\pi/15, \pi/15]\,\mathrm{rad}$, $\psi = \pi\,\mathrm{rad} \rightarrow \dot{v}_x = 0\,\mathrm{m\cdot s^{-2}}$.
3) With a zero velocity along the x-axis, zero roll, yaw orienting the drone in the positive direction of the y-axis, and a non-zero pitch, the acceleration in the direction of the x-axis has to be zero: $v_x = 0\,\mathrm{m\cdot s^{-1}}$, $\theta \in [-\pi/15, \pi/15]\,\mathrm{rad}$, $\varphi = 0\,\mathrm{rad}$, $\psi = \pi/2\,\mathrm{rad} \rightarrow \dot{v}_x = 0\,\mathrm{m\cdot s^{-2}}$.
4) With a zero velocity along the x-axis, zero roll, yaw orienting the drone in the negative direction of the y-axis, and a non-zero pitch, the acceleration in the direction of the x-axis has to be zero: $v_x = 0\,\mathrm{m\cdot s^{-1}}$, $\theta \in [-\pi/15, \pi/15]\,\mathrm{rad}$, $\varphi = 0\,\mathrm{rad}$, $\psi = -\pi/2\,\mathrm{rad} \rightarrow \dot{v}_x = 0\,\mathrm{m\cdot s^{-2}}$.

Analogously, we define four formal constraints for modeling $\dot{v}_y$, with the exception that the roles of θ and φ are swapped. All four constraints are then expected to yield zero $\dot{v}_y$.

For both $\dot{v}_x$ and $\dot{v}_y$, we have generated 50 training constraint samples and 50 test constraint samples for each of the four constraints.

To select the best model among the models generated by SNGP with formal constraints, we consider models with e_c^{train} below $\gamma = 5\times10^{-2}\,\mathrm{m\cdot s^{-2}}$ and we choose the model with the lowest e_d^{train}.

TABLE III

MEDIAN ERROR OVER 50 RUNS OF SNGP ON THE TEST DATA SET e_d^{test} AND ON THE TEST CONSTRAINT SET e_c^{test} FOR THE DRONE EXPERIMENT.

	Scenario	Empirical model	Median e_d^{test} $(\mathrm{m\cdot s^{-2}})$	Median e_c^{test} $(\mathrm{m\cdot s^{-2}})$
$\hat{\dot{v}}_x$	Baseline	Not included	7.508×10^{-1}	2.309×10^{3}
		1 prior feature	6.877×10^{-1}	2.804×10^{3}
		3 prior features	7.237×10^{-1}	8.125×10^{2}
	Constrained	Not included	1.153×10^{0}	3.207×10^{-2}
		1 prior feature	2.245×10^{-1}	4.385×10^{-2}
		3 prior features	1.980×10^{-1}	4.269×10^{-2}
$\hat{\dot{v}}_y$	Baseline	Not included	6.803×10^{-1}	2.266×10^{2}
		1 prior feature	8.536×10^{-1}	1.899×10^{3}
		3 prior features	8.260×10^{-1}	5.282×10^{2}
	Constrained	Not included	1.987×10^{-1}	3.986×10^{-2}
		1 prior feature	1.727×10^{-1}	4.643×10^{-2}
		3 prior features	1.639×10^{-1}	4.439×10^{-2}

3) Results and Discussion: The results for zero, one, and three prior features for SR without and with formal constraints and for both modeled variables are summarized in Table III. In three of four scenarios, the variants with prior features outperform the variant with no prior feature in terms of the median e_d^{test}. In the baseline SNGP for $\hat{\dot{v}}_y$, the variants with prior features perform slightly worse than the variant without the prior feature. However, the constrained SNGP clearly benefits from including the prior features. The variant with three prior features outperforms all other variants in the constrained SNGP scenarios for both variables. This result indicates that splitting the empirical model into more prior features is beneficial in certain cases.

There is no statistical difference at the significance level of 5 % between the data fitting performance e_d^{test} of the baseline SNGP with prior features and without them. However, incorporating the prior features into SNGP with formal constraints leads to significantly better results. In the case of $\hat{\dot{v}}_x$, the improvement is substantial: the Wilcoxon rank-sum test returns $p = 2.383\times10^{-9}$ for the case with one prior feature and $p = 1.520\times10^{-10}$ for the case with three prior features, both compared to the case without any prior feature. Also for $\hat{\dot{v}}_y$, a significant improvement is achieved with $p = 2.284\times10^{-3}$ for one prior feature and $p = 1.040\times10^{-5}$ for three prior features, compared to the case with no prior feature.

Similarly as for the mobile robot, it can be clearly seen that the constraint satisfaction error e_c^{test} on the test constraint samples is by several orders of magnitude better for the models learned by SNGP with formal constraints.

V. CONCLUSIONS

We have proposed a method for robot model learning based on symbolic regression that allows to incorporate prior knowledge to the model search and use it along with the training data to evolve an accurate robot model in the form of analytic equations. The prior knowledge is given to the method by specifying partially known model components,

such as theoretical or empirical models, in the form of prior features. An advantage is the ability to learn from small batches of data and combining the prior features with formal constraints.

The experimental evaluation on two systems from the robotics domain has shown that including the prior knowledge improves the model accuracy and the compliance with the physical limitations of the robot, as compared to the baseline SNGP. An overall improvement in data fitting accuracy was achieved by including the prior features both for the baseline SNGP and for SNGP with constraints.

In the future work, we will evaluate the method on higher-dimensional problems and test the performance of the models in a control loop. We will also evaluate the impact of the training set size on the method performance. Further research will also involve structure and parameter tuning within the prior features. In addition, we aim at conducting a thorough comparison of the proposed approach with alternative model learning methods.

REFERENCES

[1] O. Sigaud, C. Salaün, and V. Padois, "On-line regression algorithms for learning mechanical models of robots: a survey," *Robotics and Autonomous Systems*, vol. 59, no. 12, pp. 1115–1129, 2011.

[2] D. Nguyen-Tuong and J. Peters, "Model learning for robot control: a survey," *Cognitive Processing*, vol. 12, no. 4, pp. 319–340, Nov 2011. [Online]. Available: https://doi.org/10.1007/s10339-011-0404-1

[3] F. Stulp and O. Sigaud, "Many regression algorithms, one unified model: A review," *Neural Networks*, vol. 69, pp. 60 – 79, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0893608015001185

[4] 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.

[5] S. Vijayakumar and S. Schaal, "Locally weighted projection regression: An O(n) algorithm for incremental real time learning in high dimensional space," in *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000)*, vol. 1, 2000, pp. 288–293.

[6] G. Yehuda, M. Gabel, and A. Schuster, "It's not what machines can learn, it's what we cannot teach," *arXiv preprint arXiv:2002.09398*, 2020.

[7] M. Schmidt and H. Lipson, "Distilling free-form natural laws from experimental data," *Science*, vol. 324, no. 5923, pp. 81–85, 2009.

[8] E. Vladislavleva, T. Friedrich, F. Neumann, and M. Wagner, "Predicting the energy output of wind farms based on weather data: Important variables and their correlation," *Renewable Energy*, vol. 50, pp. 236–243, 2013.

[9] E. Derner, J. Kubalík, N. Ancona, and R. Babuška, "Constructing parsimonious analytic models for dynamic systems via symbolic regression," *Applied Soft Computing*, vol. 94, p. 106432, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1568494620303720

[10] J. Kubalík, E. Derner, and R. Babuška, "Symbolic regression driven by training data and prior knowledge," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO '20)*. New York, NY, USA: Association for Computing Machinery, 2020, pp. 958–966. [Online]. Available: https://doi.org/10.1145/3377930.3390152

[11] D. Jackson, "Single node genetic programming on problems with side effects," in *Parallel Problem Solving from Nature - PPSN XII: 12th International Conference, Taormina, Italy, September 1-5, 2012, Proceedings, Part I*, C. A. C. Coello, V. Cutello, K. Deb, S. Forrest, G. Nicosia, and M. Pavone, Eds. Berlin, Heidelberg: Springer, 2012, pp. 327–336.

[12] D. Khandelwal, M. Schoukens, and R. Toth, "Data-driven modelling of dynamical systems using tree adjoining grammar and genetic programming," in *2019 IEEE Congress on Evolutionary Computation, CEC 2019 – Proceedings*. United States: Institute of Electrical and Electronics Engineers, 6 2019, pp. 2673–2680.

[13] I. Błądek and K. Krawiec, "Solving symbolic regression problems with formal constraints," in *Proceedings of the Genetic and Evolutionary Computation Conference*. New York, NY, USA: ACM, 2019, pp. 977–984. [Online]. Available: http://doi.acm.org/10.1145/3321707.3321743

[14] S.-M. Udrescu and M. Tegmark, "AI Feynman: A physics-inspired method for symbolic regression," *Science Advances*, vol. 6, no. 16, p. eaay2631, 2020.

[15] S. Greydanus, M. Dzamba, and J. Yosinski, "Hamiltonian neural networks," in *Advances in Neural Information Processing Systems*, 2019, pp. 15 379–15 389.

[16] J. Kocijan, *Modelling and control of dynamic systems using Gaussian process models*. Springer, 2016.

[17] D. Nguyen-Tuong, M. Seeger, and J. Peters, "Model learning with local Gaussian process regression," *Advanced Robotics*, vol. 23, no. 15, pp. 2015–2034, 2009.

[18] D. Bruder, C. D. Remy, and R. Vasudevan, "Nonlinear system identification of soft robot dynamics using Koopman operator theory," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 6244–6250.

[19] A. Mauroy and J. Goncalves, "Linear identification of nonlinear systems: A lifting technique based on the Koopman operator," in *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE, 2016, pp. 6500–6505.

[20] I. Abraham and T. D. Murphey, "Active learning of dynamics for data-driven control using Koopman operators," *IEEE Transactions on Robotics*, vol. 35, no. 5, pp. 1071–1083, 2019.

[21] R. Murray-Smith and T. A. Johansen, Eds., *Multiple Model Approaches to Nonlinear Modeling and Control*. London, UK: Taylor & Francis, 1997.

[22] T. A. Johansen and B. A. Foss, "ORBIT – operating-regime-based modeling and identification toolkit," *Control Engineering Practice*, vol. 6, no. 10, pp. 1277–1286, 1998.

[23] I. Arnaldo, K. Krawiec, and U.-M. O'Reilly, "Multiple regression genetic programming," in *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, 2014, pp. 879–886.

[24] J. Kubalík, E. Derner, and R. Babuška, "Enhanced symbolic regression through local variable transformations," in *Proceedings of the 9th International Joint Conference on Computational Intelligence*, vol. 1, 2017, pp. 91–100.

[25] E. Derner, J. Kubalík, and R. Babuška, "Data-driven construction of symbolic process models for reinforcement learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 1–8.

[26] J. D. Gibbons and S. Chakraborti, *Nonparametric Statistical Inference: Revised and Expanded*. CRC press, 2014.