CZECH TECHNICAL UNIVERSITY IN PRAGUE
FACULTY OF ELECTRICAL ENGINEERING
DEPARTMENT OF CONTROL ENGINEERING

Doctoral Thesis

DATA-EFFICIENT METHODS FOR MODEL LEARNING AND CONTROL IN ROBOTICS

by **Ing. Erik Derner**

presented to the Faculty of Electrical Engineering,
Czech Technical University in Prague,
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy (Ph.D.)

Supervisor: Prof. Dr. Ing. Robert Babuška
Study program: Electrical Engineering and Information Technology
Branch of study: Control Engineering and Robotics
February 2022

Contents

SUMMARY

Constructing mathematical models of dynamic systems is central to many engineering and science disciplines. Models facilitate simulations, analysis of the system's behavior, decision making, and design of automatic control algorithms. Even inherently model-free control techniques such as reinforcement learning have been shown to benefit from the use of models. However, applying model learning methods to robotics is not straightforward. Obtaining informative data for constructing dynamic models can be difficult, especially when the models are to be learned during task execution. Despite their increasing popularity, commonly used model learning methods such as deep neural networks come with drawbacks. They are data-hungry and require a lot of computational power to learn a large number of parameters in their complex structure. Their black-box nature does not offer any insight into or interpretation of the model. Also, configuring these methods to achieve good results is often a difficult task.

The objective of this thesis is to address the present challenges in data-driven model learning in robotics. Several variants and extensions of symbolic regression are introduced. This technique, based on genetic programming, is suitable to automatically build compact and accurate models in the form of analytic equations even from small data sets. One of the challenges is posed by the large amount of data the robots collect during their operation, demanding techniques to select a smaller subset of training samples. To that end, this thesis presents a novel sample-selection method based on model prediction error and compares it to four alternative approaches. A real-world experimental evaluation on a mobile robot shows that a model learned from only a few tens of samples selected by the proposed method can be used to accomplish a motion control task within a reinforcement learning scheme.

Standard data-driven model learning techniques in many cases yield models that violate the physical constraints of the robot. However, a partial theoretical or empirical model of the robot is often known. It is shown in this work how symbolic regression can be naturally extended to include the prior information into the model construction process. An experimental evaluation on two real-world robotic platforms demonstrates that symbolic regression is able to automatically build models that are both accurate and physically valid and compensate for theoretical or empirical model deficiencies.

Efficient methods are needed not only to learn robot models but also to learn models of the robot's environment. The thesis is concluded by presenting a novel method for reliable robot localization in dynamic environments. The proposed approach introduces an environment representation based on weighted local visual features and a change detection algorithm that updates the weights as the robot moves around the environment. The core idea of the method consists in using the weights to distinguish the useful information in stable regions of the scene from the unreliable information in the regions that are changing. An extensive evaluation and comparison to state-of-the-art alternatives show that using the proposed change detection algorithm improves the localization accuracy.

Keywords: robotics, model learning, symbolic regression, genetic programming, reinforcement learning, robot control, localization.

ANOTACE

Znalost matematických modelů dynamických systémů je klíčová pro celou řadu inženýrských a vědeckých disciplín. Modely umožňují provádění simulací, analýzu chování systému, rozhodování a návrh řídicích algoritmů. Z použití modelů těží i techniky, které z principu fungují bez modelu, například posilované učení. Využití metod pro učení modelů v robotice má však svá specifika. Získat informativní data pro učení dynamických modelů může být obtížné, zvláště během vykonávání dané úlohy. Navzdory rostoucí popularitě mají běžně používané metody učení modelů, jako jsou hluboké neuronové sítě, své nevýhody. Vyžadují velký objem trénovacích dat a značný výpočetní výkon, aby se naučily velký počet parametrů. Jejich black-box charakter neumožňuje interpretaci modelu ani vhled do jeho struktury. Také správné nastavení konfigurace pro dosažení dobrých výsledků je u těchto metod často obtížný úkol.

Cílem této disertační práce je navrhnout řešení aktuálních problémů v oblasti učení modelů z dat v robotice. Práce představuje několik variant a rozšíření symbolické regrese. Tato technika, založená na genetickém programování, je vhodná pro automatické vytváření kompaktních a přesných modelů v podobě analytických rovnic i z malých souborů dat. Jedním z problémů v robotice je velké množství dat, které jsou roboty během provozu shromažďovány, což vyžaduje výběr podmnožiny trénovacích vzorků. Tato práce představuje novou metodu výběru vzorků založenou na predikční chybě modelu a porovnává ji se čtyřmi alternativními metodami. Experimentální vyhodnocení na mobilním robotu ukazuje, že model naučený jen z několika desítek vzorků vybraných navrženou metodou může být využit pro úspěšné vykonání úlohy založené na řízení metodou posilovaného učení.

Běžně používané techniky učení modelů z dat v mnoha případech generují modely, které nevyhovují fyzikálním omezením robota. Částečný teoretický nebo empirický model robota je přitom často znám. Tato práce ukazuje, jak lze symbolickou regresi přirozeně rozšířit tak, aby byly předem známé informace o robotu zahrnuty do procesu učení modelu. Experimentální vyhodnocení na dvou různých robotech ukazuje, že symbolická regrese je schopna automaticky vytvářet modely, které jsou přesné, vyhovují fyzikálním omezením a kompenzují nedostatky teoretického nebo empirického modelu.

Efektivní metody jsou třeba nejen k učení modelů robotů, ale také k učení modelů prostředí robota. Práce je zakončena představením nové metody pro spolehlivou lokalizaci robotů v dynamických prostředích. V navrhovaném přístupu se využívá model prostředí založený na vážených lokálních vizuálních příznacích. Algoritmus detekce změn průběžně aktualizuje tyto váhy během pohybu robota prostředím. Základní myšlenkou metody je na základě těchto vah rozlišit užitečné informace ve stabilních oblastech scény od nespolehlivých informací v oblastech, které se mění. Rozsáhlé experimentální vyhodnocení a srovnání s alternativními metodami ukazuje, že použití navrženého algoritmu detekce změn zlepšuje přesnost lokalizace.

Klíčová slova: robotika, učení modelů, symbolická regrese, genetické programování, posilované učení, řízení robotů, lokalizace.

ACKNOWLEDGMENTS

I have been honored to be accompanied by many great people on my journey of pursuing a doctoral degree. Without them, finishing this thesis would not have been possible. In the following paragraphs, I want to express my gratitude to these valuable and inspiring people.

First of all, I would like to sincerely thank my supervisor, Robert Babuška, for his continuous support, invaluable advice, and patience. Since the beginning, I have felt like a team member rather than a student or an employee. Not only his exceptional research skills but also his leadership skills based on mutual respect and trust have always been a great inspiration to me. I could not have wished for a better supervisor. Thank you so much for believing in me and giving me the opportunity to pursue my Ph.D. under your supervision.

I have been fortunate to have Jiří Kubalík as my closest colleague, mentor, and friend. He had introduced me to the exciting world of biologically inspired algorithms already during my Master's studies. Actually, it was him who had made me join the research team and pursue a Ph.D. degree in this field. He has always been a source of friendly atmosphere, making me feel comfortable asking whatever question. Many thanks for motivating me to go on even in the most challenging times through your positive attitude and optimism.

From my colleagues, I would definitely like to mention Jonáš Kulhánek. He joined our team to work on his Bachelor's thesis and I was glad to see him quickly grow into a great researcher thanks to his exceptional skills and contagious enthusiasm. I have learned a lot during our discussions and working on common projects.

An integral part of my research was carried out in collaboration with the Robotics Lab at the University Carlos III of Madrid. I would like to thank Ramón Barber for making this collaboration possible by welcoming me for a couple of research stays in his team. During this collaboration, I was privileged to work together with Clara and Alejandra on research that resulted in several joint publications. They have been not only talented colleagues to me but also great amigas.

I would also like to thank the administrative staff, namely Svatava, Johana, Martin, and Jana, for dealing with various requests I had. Thank you all for being so helpful and kind. I really appreciate it as it is something that should not be taken for granted.

Throughout my studies, I have spent most of my free time with the International Student Club CTU in Prague. This amazing entity, a group of people connected through an inexpressible spirit, has been my second family and therefore deserves its place in the acknowledgments. In particular, I would like to mention Evča, Terez, Michal, and Carmen. You have been a great inspiration and support to me throughout the past years.

The list would never be complete without giving thanks to Zdenča, who, despite the challenges she had to face in her own life, has provided me with unconditional support in the tough last months. Our endless deep conversations helped me overcome all the troubles and carry on. I am really glad to have you in my life.

I could not conclude this text in another way than by expressing sincere gratitude to my parents. Their boundless support, motivation, and care have driven me throughout my studies while I never had to worry about having a place to go whenever I needed. My mom had supported me to go for a Ph.D. when I was considering what to do next after finishing my Master's degree. She has always been interested in whether I am doing well, and I am really grateful for this.

In life, people come and go, and the period of my doctoral studies was not an exception. Unfortunately, some people that had a special place in my life no longer form a part of it. At this place, I would like to thank Anahí, who had been by my side for a large part of my studies. Even though this is no longer the case when I am finishing my thesis, I am grateful for all the great moments we spent together.

Finally, let me say a couple of words in honor of my dad. It leaves me with a lot of sadness that he will not be able to read this thesis. He was a positive and inspiring person, the role model of my life. He always strived to make people around him happy, and he also gave true meaning to the definition of reliability. He lost the fight against an insidious disease in 2020. I would like to dedicate this thesis to him.

1 Introduction

In robotics, many algorithms rely on an accurate model of the system. Model-based techniques comprise a wide variety of methods such as model predictive control, time series prediction, fault detection and diagnosis, or reinforcement learning (RL). While model-free algorithms are available, the absence of a model has many consequences; from increasing the risk of damage for the robot to slowing down the convergence of learning algorithms. Therefore, using model-free methods remains an open topic in robotics, requiring pre-training using a simulator. Even though using robot models can be advantageous, it comes with certain limitations and challenges, detailed in Section 1.1. This thesis addresses a number of these challenges, as explained in Section 1.2.

1.1. Motivation and Challenges

Physical (first-principle) models are readily available for the robot mechanics, such as robot arms or mobile ground robots [6, 129]. They are usually given in an analytic form, which makes them transparent and intuitive, allowing for an insight into the model structure. However, using physical models comes with considerable limitations. Such models are commonly simplified and do not capture all properties of the real system. Certain phenomena are difficult to model, for instance, contact dynamics, grasping of deformable objects, aerodynamic phenomena, friction, etc. This yields physical models incomplete, not describing all the parts that eventually play an important role in controlling the robot. Moreover, the parameters of physical models are difficult to optimize. Optimization through ad-hoc methods requires a large amount of data and complete state measurements, which are commonly not available. Finally, physical models cannot be used to represent the robot's environment, as it is usually unknown at the time the robot is designed.

1.1.1. Challenges in Data-Driven Model Learning

To overcome the limitations of physical models, data-driven model learning techniques can be employed to learn models of the robot's dynamics using training samples collected from the robot. An advantage of learning robot models from data is that it allows capturing all the properties of a particular real robot, which are reflected in the data. Many model-learning approaches have been developed over the past decades and applied in robotics: time-varying linear models [95, 147], probabilistic models such as Gaussian processes [40, 146], basis function expansions [13, 108], regression trees [50, 132], deep neural networks [60, 106], or local linear regression [10, 49].

Most of these methods need to learn a large number of parameters. For instance, deep neural networks may need to learn hundreds of thousands of weights in the network structure. Learning a lot of parameters requires a lot of training samples to obtain an accurate model. Moreover, complex structures and the high representative power of these models often lead to overfitting, which substantially decreases the performance of the model on previously unseen data. In relation to overfitting, data-driven model learning methods exhibit a tendency to produce models that perform poorly when extrapolating. These methods are also sensitive to an appropriate configuration, known as hyperparameter tuning in the case of deep neural networks. Learning such models comes at the cost of high computational complexity, which leads to extensive training times. To make the training feasible, expensive hardware such as efficient graphics cards is needed. Finally, many of the aforementioned model-learning methods yield black-box models. Such models are not interpretable and do not offer much insight.

1.1.2. MODEL LEARNING CHALLENGES IN ROBOTICS

To apply data-driven model learning approaches to robotics, several challenges need to be overcome [122]. Techniques for learning models based on data 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.

Data collection is expensive and potentially unsafe, in particular when applying random control inputs to the robot. It is preferred to collect data during standard robot operation, using safe control inputs. However, such data are often biased, as they include only a restricted subset of the state-action space from the problem domain.

The amount of data samples that the robot continuously collects during its deployment quickly grows in time, which poses a challenge for model-learning algorithms. As the training set gets too large, using all collected data samples becomes computationally infeasible. To remedy this issue, a subset of the data must be selected for model learning. However, not all data samples are equally important, and it is not known a priori which samples will be useful. Moreover, samples from repetitive motions, which prevail in many robotics tasks, often outweigh other, less frequent but informative samples.

Another important consideration to be taken into account is that a robot model needs to respect the physical constraints of the robot. Common model-learning methods produce models that violate the physical constraints, such as non-holonomic constraints in mobile robots. A controller based on such a model may exploit these model deficiencies and lead to meaningless control results, such as a wheeled unicycle robot moving sideways.

Data-efficient methods are needed not only to learn the model of the robot's dynamics, but also to learn the model of the robot's environment. Deployment of autonomous mobile robots in industrial and domestic environments is challenging due to the dynamics of these environments. Examples of changes in such environments include moving chairs and items on tables, altering pictures on computer or TV screens, changing contents of whiteboards and notice boards, opening or closing doors, adjusting blinds in windows, etc. Such changes happen often and cause many conventional localization and place detection methods to fail. Advanced methods exploiting the information about the changes are needed to perform localization and navigation precisely and reliably, despite the changes occurring in the environment.

1.2. Objectives and Contributions

The objective of this thesis is to address the aforementioned challenges in learning the model of the robot and its environment. To deal with the challenges in data-driven model learning, the thesis introduces several variants and extensions of *symbolic regression* (SR). Symbolic regression is a technique based on genetic programming. It evolves tree-like structures representing mathematical expressions in a manner resembling natural evolution, using genetic operators such as mutation. Given a set of training data samples, it automatically finds models in the form of analytic equations. Similarly as for physical models, the analytic form of the models constructed by means of SR allows for their further analysis and facilitates plugging them into other algorithms. Contrary to deep neural networks, SR learns accurate, parsimonious models even from very small data sets. The resulting models are typically by several orders of magnitude smaller, in terms of the number of parameters, than models represented by deep neural networks. Moreover, SR does not suffer from the exponential growth of the model complexity with the dimensionality of the problem.

1.2.1. Symbolic Regression for Robot Model Learning

Chapter 2 gives a general introduction to SR-based model learning and presents the SR methods used in this thesis. Furthermore, it introduces the RL control framework used in the experiments. Chapter 3, based on [43], shows how SR can be applied to find accurate and compact robot models for systems of varying dimensionality: from a one-degree-of-freedom robot arm through a mobile robot to a complex bipedal walking robot system. It shows SR as a powerful tool that overcomes the drawbacks of alternative methods such as deep neural networks. Using SR reduces the number of training samples and the model complexity by several orders of magnitude.

1.2.2. Informative Training Sets

To deal with the large amount of data obtained during robot operation and the bias that is often present in such data collections, a suitable subset of the data needs to be selected for model learning. Approaches to selecting informative samples from large data collections are discussed in Chapter 4, based on [47]. Intuitively, samples can be selected sequentially or at random. Other techniques include choosing the training samples based on certain criteria imposed on the data or intermediate models. Therefore, data samples can be chosen to evenly cover the problem domain [128]. Alternatively, techniques based on model disagreement measured by the model output variance [20] can be employed. Next to these methods from the literature, this thesis introduces a novel approach based on the model prediction error, overcoming certain limitations of the available techniques. A comparison of various approaches to sample selection is demonstrated on a framework using SR for model learning. As SR can be time-consuming for large data sets, selecting a suitable small training set of informative samples makes it very well usable in practice.

1.2.3. Model Learning with Prior Knowledge

Standard SR may produce models that are accurate, but do not comply with the physics, such as with the non-holonomic constraints of a robot. A partial information about the robot model is often known, such as a theoretical or empirical model, and can be exploited in model learning. In its standard form, SR allows to incorporate the prior knowledge about the robot (system) into the model construction process by specifying the set of elementary functions that can be used to build the analytic models. However, prior knowledge about the particular robot can be included also in the form of formal constraints or partial models. Chapter 5, based on [46], shows that SR allows to naturally incorporate the prior knowledge into the model construction process. Using formal constraints leads to a multi-objective optimization based on two optimization criteria: the root-mean-square error on the training data and the formal constraint violation error. The knowledge of prior physical or empirical models (or their parts) allows to include them in the form of building blocks 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.

1.2.4. Efficient Robot Localization in Dynamic Environments

With the increasing availability of affordable hardware components including cameras, many vision-based methods can be applied in robotics. One of the main challenges is the ability of robot self-localization in dynamic environments. In the last part of the thesis, Chapter 6, a novel approach to change detection based on local features and descriptors [42] is introduced. A mobile robot is equipped with a camera as its only sensor. The robot moves through its environment and detects changes that have occurred with respect to an initial mapping session. Once the robot detects a change, it captures the change by decreasing weights assigned to the corresponding features stored in the environment representation. By contrast, stable features that have been detected during multiple visits to the same place are given higher importance through increasing their weights. The method is evaluated on public data as well as on our own long-term localization data set and compared to alternative methods from the literature. The advantages of the proposed method are its fast learning, low computational resources demand, data efficiency, and low requirements on the sensor hardware.

2
Model Learning Preliminaries

This chapter gives a general introduction to symbolic regression (SR) in Section 2.1. A variant of SR used in this thesis, Single Node Genetic Programming (SNGP), is explained in Section 2.2. An extension to SR that allows to include prior knowledge in the form of constrains into the model search process is described in Section 2.3. Section 2.4 defines nonlinear dynamic system models and explains how SR is applied to automatically construct them from data. Section 2.5 provides preliminaries of model-based reinforcement learning (RL), which is used for robot control in the experiments.

2.1. Symbolic Regression

Symbolic regression is a type of regression analysis that searches the space of analytic expressions to find a model that fits the best the given training data set. The structure of the model is not given in advance. Instead, it is learned automatically together with its parameters. This makes the search space very large, which leads to extensive learning times. Similarly as for other regression methods, the objective function is defined as an error observed on the training data. A typical example of an objective function is the root-mean-square error (RMSE).

SR is commonly performed through genetic programming (GP). GP is an evolutionary-based metaheuristic that typically operates on candidate solutions represented by tree-like structures, which is a natural way to encode analytic expressions. GP does not require the model structure to be specified beforehand; it only needs the set of elementary functions used to construct the analytic expressions. Also, parameters defining upper bounds on the model complexity are to be given in order to prevent generating excessively large models. An advantage of GP is that it can find accurate solutions in a reasonable time. Another benefit of applying GP to solve the SR problem is that it works with a population of candidate solutions, yielding a pool of well-fit models at the end of the optimization process. A schematic overview of SR solved by means of GP is shown in Figure 2.1. Many GP variants have been developed, including the original tree-based GP [84], and others, such as Grammatical Evolution [124], Cartesian GP [105], Gene Expression Programming [55], Single Node Genetic Programming (SNGP) [70, 71], and Multi-Gene Genetic Programming (MGGP) [127, 145]. The last two variants are used in this work to learn robot models from data.

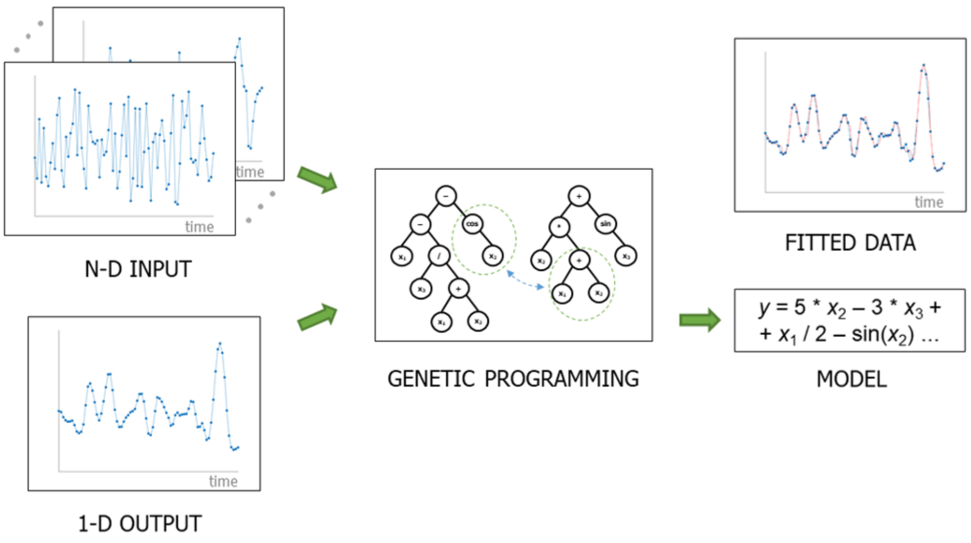
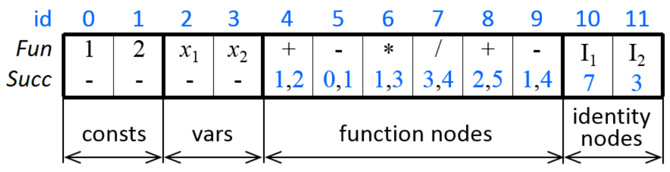


Figure 2.1: Overview of the symbolic regression algorithm principle. The population of individuals is represented by a set of tree-like structures encoding mathematical expressions. The algorithm automatically evolves a model in the form of an analytic expression by successively applying genetic operators to the population to fit a data set of N-dimensional inputs and one-dimensional outputs as accurately as possible.

2.2. SINGLE NODE GENETIC PROGRAMMING

The baseline SR algorithm used in this thesis is based on Single Node Genetic Programming (SNGP) [70, 71]. It is a graph-based GP technique evolving a population of individuals organized in a linear array, where each array element represents a single program node, see Figure 2.2.



id	0	1	2	3	4	5	6	7	8	9	10	11
Fun	1	2	x_1	x_2	+	-	*	/	+	-	I_1	I_2
Succ	-	-	-	-	1,2	0,1	1,3	3,4	2,5	1,4	7	3
	consts		vars		function nodes						identity nodes	

Figure 2.2: A simplified structure of the standard SNGP population [87], consisting of constants, variables, function nodes, and identity nodes. Each element of the array represents a program node. Each program node has an associated function (including constants and variables) and link(s) to its successor(s).

Each function node represents a root of an expression that can be constructed recursively by traversing its successors. It can only take nodes to its left in the population as its successors, i.e., operands. Constants and variables are leaf nodes. For example, the node with id = 4 represents addition of the outputs of the nodes with id = 1 and id = 2, forming the mathematical expression $2 + x_1$.

The final model is a weighted sum of multiple features. These features are expressions rooted in the nodes that are successors of the identity nodes. In the example given in Figure 2.2, two features are used: the first feature is the expression rooted at the node with id = 7 and the second one is the expression rooted at the node with id = 3. The features are formed through an evolutionary process, while their coefficients are found by linear least squares, see Section 2.4.

The main parameters of SNGP are:

- **Population size** – the total number of individuals in the population.
- **Maximum number of features** – the number of identity nodes.
- **Maximum expression tree depth** – the maximum number of levels between the expression root and its deepest leaf node. This upper bound is applied to each function node in the population.
- **Maximum number of generations** – the number of iterations of the evolutionary process.
- **Elementary function set** – the set of functions used in the function nodes, e.g., addition, multiplication, square, sine, cosine.

The evolution is an iterative process where in each iteration, a new population is derived from the current one by means of a single operator called *successor mutation*. In each iteration, one node in the population is selected either randomly or using some selection operator. Then, one successor of the node is replaced by a randomly chosen new one. This operation must not violate the constraints imposed on the maximum expression depth and the SNGP rule that the successor nodes need to be to the left of a given node. The new population will replace the current one if and only if the mutation operation did not worsen the model's fitness. For a more detailed description of the SNGP variant used in this work please refer to [87].

2.3. Symbolic Regression with Formal Constraints

Standard SR optimizes the model according to a single objective, typically the RMSE on the training data set. It yields models that are accurate with respect to the training data, but often ignore the physical constraints of the system (robot). However, prior knowledge about the system is often available. This prior knowledge captures important high-level characteristics of the system's physical laws, without requiring deep and exact knowledge of the physical model. It can be expressed as formal constraints imposed on the model parameters or function values, but also as a description of the steady-state behavior of the robot, as the velocity and acceleration trends under certain inputs, etc.

To that end, [89] proposes a SR extension to use bi-objective optimization that finds models fitting well the training data and complying with the specified constraints at the same time. The constraints are represented by auxiliary data samples capturing the desired behavior. In this way, the extent to which the candidate models violate the constraints can be quantified. The two optimization criteria are formally defined by the following objectives:

- Minimize the RMSE on the training data set. The training data set consists of data tuples in the form $(\mathbf{x}_k, \mathbf{u}_k, \mathbf{x}_{k+1})$, where $\mathbf{x}_k$ represents the current state vector, $\mathbf{u}_k$ is the current input, and $\mathbf{x}_{k+1}$ denotes the next state vector at the next time step after applying the input $\mathbf{u}_k$ (see also Section 2.4).

- Minimize the RMSE on the training constraint set. This measure captures how well the model complies with the prior knowledge through the error on the auxiliary data samples generated for the user-specified formal constraints.

The multi-objective SNGP proposed in [88] works with a population of models, where each individual is a model represented by a baseline SNGP population. The population of models is evolved using the domination-based multi-objective evolutionary algorithm NSGA-II [39]. The *domination* principle is defined as follows: A solution s_1 dominates another solution s_2 if s_1 is not worse than s_2 in either objective and s_1 is strictly better than s_2 in at least one objective. For more details on the bi-objective SR with formal constraints please refer to [89].

2.4. NONLINEAR DYNAMIC SYSTEM MODEL

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

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) \tag{2.1}$$

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 work, 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 $j = 1, \ldots, n$:

$$\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) = \left(f^1(\mathbf{x}_k, \mathbf{u}_k), f^2(\mathbf{x}_k, \mathbf{u}_k), \ldots, f^n(\mathbf{x}_k, \mathbf{u}_k)\right)^\top. \tag{2.2}$$

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 SNGP to form the function $f(\mathbf{x}_k, \mathbf{u}_k)$ for each state variable as a linear combination of evolved nonlinear functions $f_i(\mathbf{x}_k, \mathbf{u}_k)$:

$$f(\mathbf{x}_k, \mathbf{u}_k) = \beta_0 + \sum_{i=1}^{n_f} \beta_i f_i(\mathbf{x}_k, \mathbf{u}_k). \tag{2.3}$$

The SNGP algorithm builds the functions $f_i(\mathbf{x}_k, \mathbf{u}_k)$, called features, from a user-defined set of elementary functions $\mathscr{F}$. 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 maximum number of features n_f and the maximum feature depth d. To perform bi-objective optimization taking into account also prior knowledge in the form of formal constraints, as explained in Section 2.3, NSGA-II [39] is applied to a population of individuals represented by the baseline SNGP described here.

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 outputs and inputs, see Section 3.2 and [45].

2.5. REINFORCEMENT LEARNING

The system for which an optimal control strategy is to be learned is described by a nonlinear state-space model (2.1) or by a nonlinear input–output model. In the sequel, we describe reinforcement learning for the state-space model.

The *reward function* assigns a scalar reward $r_{k+1} \in \mathbb{R}$ to the state transition from $\mathbf{x}_k$ to $\mathbf{x}_{k+1}$, under action $\mathbf{u}_k$:

$$r_{k+1} = \rho(\mathbf{x}_k, \mathbf{u}_k, \mathbf{x}_{k+1}) \,. \tag{2.4}$$

The reward function ρ specifies the control goal, typically as the distance of the current state to a given goal state.

Based on the model (2.1), we compute the optimal control policy $\pi : \mathcal{X} \to \mathcal{U}$ such that in each state it selects a control action so that the expected cumulative discounted reward over time, called the return, is maximized:

$$R^\pi = E\Big\{\sum_{k=0}^{\infty} \gamma^k \rho\big(\mathbf{x}_k, \pi(\mathbf{x}_k), \mathbf{x}_{k+1}\big)\Big\} \,. \tag{2.5}$$

Here $\gamma \in (0,1)$ is a discount factor and the initial state $\mathbf{x}_0$ is drawn from the state space domain $\mathcal{X}$ or its subset. Over the whole state space, the return is captured by the value function $V^\pi : \mathcal{X} \to \mathbb{R}$ defined as:

$$V^\pi(\mathbf{x}) = E\Big\{\sum_{k=0}^{\infty} \gamma^k \rho\big(\mathbf{x}_k, \pi(\mathbf{x}_k), \mathbf{x}_{k+1}\big)\Big| \mathbf{x}_0 = \mathbf{x}\Big\} \,. \tag{2.6}$$

An approximation of the optimal V-function, denoted by $\hat{V}^*(\mathbf{x})$, can be computed by solving the Bellman optimality equation

$$\hat{V}^*(\mathbf{x}) = \max_{\mathbf{u} \in \mathcal{U}} \Big[\rho\big(\mathbf{x}, \mathbf{u}, \mathbf{f}(\mathbf{x}, \mathbf{u})\big) + \gamma \hat{V}^*\big(\mathbf{f}(\mathbf{x}, \mathbf{u})\big)\Big] \,. \tag{2.7}$$

In this thesis, the function $\mathbf{f}(\mathbf{x}, \mathbf{u})$ is represented by the model found by means of SR.

To simplify the notation, we drop the superscripts; $V(\mathbf{x})$ therefore denotes an approximation of the optimal V-function. Based on $V(\mathbf{x})$, the corresponding approximately optimal control action is found as the one that maximizes the right-hand side of (2.7):

$$\mathbf{u} = \underset{\mathbf{u}' \in U}{\text{argmax}} \left[\rho(\mathbf{x}, \mathbf{u}', \mathbf{f}(\mathbf{x}, \mathbf{u}')) + \gamma V(\mathbf{f}(\mathbf{x}, \mathbf{u}'))\right] \,. \tag{2.8}$$

In this work, the above equation is used online as the control policy π with a set of discretized inputs U, so that the near-optimal control action can be found by enumeration.

3
Constructing Parsimonious Analytic Models for Dynamic Systems

Developing mathematical models of dynamic systems is central to many disciplines of engineering and science. Models facilitate simulations, analysis of the system's behavior, decision making and design of automatic control algorithms. Even inherently model-free control techniques such as reinforcement learning (RL) have been shown to benefit from the use of models, typically learned online. Any model construction method must address the tradeoff between the accuracy of the model and its complexity, which is difficult to strike. In this chapter, we propose to employ symbolic regression (SR) to construct parsimonious process models described by analytic equations. We have equipped our method with two different state-of-the-art SR algorithms which automatically search for equations that fit the measured data: Single Node Genetic Programming (SNGP) and Multi-Gene Genetic Programming (MGGP). In addition to the standard problem formulation in the state-space domain, we show how the method can also be applied to input–output models of the NARX (nonlinear autoregressive with exogenous input) type. We present the approach on three simulated examples with up to 14-dimensional state space: an inverted pendulum, a mobile robot, and a bipedal walking robot. A comparison with deep neural networks and local linear regression shows that SR in most cases outperforms these commonly used alternative methods. We demonstrate on a real pendulum system that the analytic model found enables an RL controller to successfully perform the swing-up task, based on a model constructed from only 100 data samples.

This chapter is an adapted version of the journal paper [43].

3.1. INTRODUCTION

Numerous methods rely on an accurate model of the system. Model-based techniques comprise a wide variety of methods such as model predictive control [104, 121], time series prediction [102], fault detection and diagnosis [61, 142], or reinforcement learning (RL) [107, 137].

Even though model-free algorithms are available, the absence of a model slows down convergence and leads to extensive learning times [65, 80, 119]. Various model-based methods have been proposed to speed up learning [58, 68, 75, 92, 136]. To that end, many model-learning approaches are available: time-varying linear models [95, 98], Gaussian processes [19, 40] and other probabilistic models [113], basis function expansions [27, 108], regression trees [50], deep neural networks [37, 38, 67, 93, 97, 106, 107] or local linear regression [10, 64, 96].

All the above approaches suffer from drawbacks induced by the use of the specific approximation technique, such as a large number of parameters (deep neural networks), local nature of the approximator (local linear regression), computational complexity (Gaussian processes), etc. In this chapter, we propose another way to capture the system dynamics: using analytic models constructed by means of the symbolic regression method (SR). Symbolic regression is based on genetic programming and it has been used in nonlinear data-driven modeling, often with quite impressive results [24, 125, 133, 144, 145].

Symbolic regression appears to be quite unknown to the machine learning community as only a few works have been reported on the use of SR for control of dynamic systems. For instance, modeling of the value function by means of genetic programming is presented in [118], where analytic descriptions of the value function are obtained based on data sampled from the optimal value function. Another example is the work [3], where SR is used to construct an analytic function, which serves as a proxy to the value function and a continuous policy can be derived from it. A multi-objective evolutionary algorithm was proposed in [23], which is based on interactive learning of the value function through inputs from the user. SR is employed to construct a smooth analytic approximation of the policy in [86], using the data sampled from the interpolated policy. To our best knowledge, there have been no reports in the literature on the use of symbolic regression for constructing a process model in model-based control methods. We argue that the use of SR for model learning is a valuable element missing from the current nonlinear control schemes and we demonstrate its usefulness.

In this chapter, we extend our previous work [44, 45], which indicated that SR is a suitable tool for this task. It does not require any basis functions defined a priori and contrary to (deep) neural networks it learns accurate, parsimonious models even from very small data sets. Symbolic regression can handle also high-dimensional problems and it does not suffer from the exponential growth of the computational complexity with the dimensionality of the problem, which we demonstrate on an enriched set of experiments including a complex bipedal walking robot system. In this work, we extend the use of the method to the class of input–output models, which are suitable in cases when the full state vector cannot be measured. By testing our method with two different state-of-the-art genetic programming algorithms, we demonstrate that the method is not dependent on the particular choice of the SR algorithm.

The chapter is organized as follows. Sections 3.2 and 3.3 present the relevant context for model learning and the proposed method. The experimental evaluation of the method is reported in Section 3.4 and the conclusions are drawn in Section 3.5.

3.2. THEORETICAL BACKGROUND

The discrete-time nonlinear state-space process model is described as

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) \tag{3.1}$$

with the state $\mathbf{x}_k, \mathbf{x}_{k+1} \in \mathscr{X} \subset \mathbb{R}^n$ and the input $\mathbf{u}_k \in \mathscr{U} \subset \mathbb{R}^m$. Note that the actual process can be stochastic (typically when the sensor readings are corrupted by noise), but in this work we aim at constructing a deterministic process model (3.1).

The full state vector cannot be directly measured for a vast majority of processes and a state estimator would have to be used. In the absence of an accurate process model, such a reconstruction is inaccurate and has a negative effect on the overall performance of the control algorithm on the real system. Note that this problem has not been explicitly addressed in the literature, as most results are demonstrated on simulation examples in which the state information is available.

Therefore, next to state-space models, we also investigate the use of dynamic input–output models of the NARX (nonlinear autoregressive with exogenous input) type. The NARX model establishes a relation between the past input–output data and the predicted output:

$$\mathbf{y}_{k+1} = \mathbf{g}\left(\mathbf{y}_k, \mathbf{y}_{k-1}, \ldots, \mathbf{y}_{k-n_y+1}, \mathbf{u}_k, \mathbf{u}_{k-1}, \ldots, \mathbf{u}_{k-n_u+1}\right), \tag{3.2}$$

where n_y and n_u are user-defined integer parameters based on the expected system's order, and $\mathbf{g}$ is a static function, different from the function $\mathbf{f}$ used in the state-space model (3.1).

For the ease of notation, we group the lagged outputs and inputs into one vector:

$$\boldsymbol{\varphi}_k = \left(\mathbf{y}_k, \mathbf{y}_{k-1} \ldots, \mathbf{y}_{k-n_y+1}, \mathbf{u}_{k-1}, \ldots, \mathbf{u}_{k-n_u+1}\right) \tag{3.3}$$

and write model (3.2) as:

$$\mathbf{y}_{k+1} = \mathbf{g}\left(\boldsymbol{\varphi}_k, \mathbf{u}_k\right). \tag{3.4}$$

Analogously to the state-space models, we will use $\mathbf{y}$ to denote the entire vector of variables and y for a single generic variable (see also Section 2.4). Note that in this setting, the model function and also the control policy are found from data samples which live in a space that is very different from the state space. The lagged outputs $y_k, y_{k-1}, \ldots, y_{k-n_y+1}$ are highly correlated and therefore span a deformed space. This presents a problem for many types of approximators. For instance, basis functions defined by the Cartesian product of the individual lagged variables will cover the whole product space $y_k \times y_{k-1} \times \ldots \times y_{k-n_y+1}$, while data samples only span a small, diagonally oriented part of the space, as illustrated in Figure 3.1. The SR approach described in this work does not suffer from such drawbacks.

In this work, we use reinforcement learning as the control method of choice. Please refer to Section 2.5 for details on the RL method used.

3.3. METHOD

In this section, we explain the principle of our method, briefly describe two variants of genetic programming algorithms used in this work, and discuss the computational complexity of our approach.

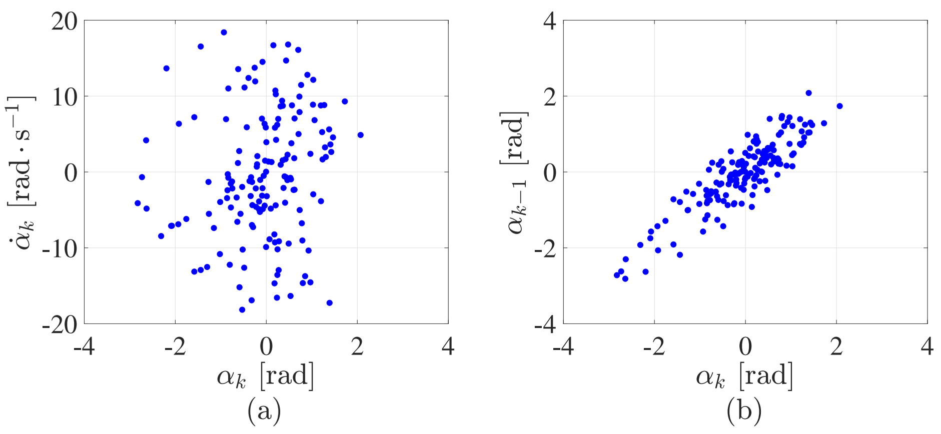


Figure 3.1: An example of trajectory samples obtained from the real inverted pendulum (see Section 3.4.3) in the original state space (a), and in the space formed by the current and previous output (b).

3.3.1. Symbolic Regression

Symbolic regression is employed to approximate the unknown state transition function f in the state-space model (3.1) or g in the input–output model (3.2). Note that each state variable is modeled individually, see Section 2.4. The analytic expressions describing the process to be controlled are constructed through genetic programming. SR methods were reported in the literature to work faster when using a linear combination of evolved nonlinear functions instead of evolving the whole analytic expression at once [7, 8]. Therefore, we define the class of analytic state-space models as:

$$f(\mathbf{x},\mathbf{u}) = \beta_0 + \sum_{i=1}^{n_f} \beta_i f_i(\mathbf{x},\mathbf{u}) \tag{3.5}$$

and the class of analytic input–output (NARX) models as:

$$g(\boldsymbol{\varphi},\mathbf{u}) = \beta_0 + \sum_{i=1}^{n_f} \beta_i g_i(\boldsymbol{\varphi},\mathbf{u})\,. \tag{3.6}$$

The nonlinear functions $f_i(\mathbf{x},\mathbf{u})$ or $g_i(\boldsymbol{\varphi},\mathbf{u})$, called features, are constructed from a set of user-defined elementary functions. These functions can be nested and are evolved by means of standard evolutionary algorithm operations, such as mutation, so that the mean-square error calculated over the training data set is minimized. No a priori knowledge on the structure of the nonlinear model is needed. The set of elementary functions may be broad to let the SR algorithm select functions that are most suitable for fitting the given data. However, it is also possible to provide the algorithm with a partial knowledge about the problem. A narrower selection of elementary functions restricts the search space and speeds up the evolution process.

To avoid over-fitting, we control the complexity of the regression model by imposing a limit on the number of features n_f and the maximum depth d of the tree representation of the features. The coefficients β_i are estimated by least squares.

3.3.2. GENETIC PROGRAMMING METHODS USED

To demonstrate that our method is not dependent on the particular choice of the SR algorithm, we test our approach with two different genetic programming methods: a modified version of Single Node Genetic Programming (SNGP) [70, 71, 87] and a modified version of Multi-Gene Genetic Programming (MGGP) [145]. Both methods have been successfully used for symbolic regression, with several applications in the RL and robotics domains [44, 45, 86, 88].

SNGP is a graph-based genetic programming technique that evolves a population of nodes organized in the form of an ordered linear array. The nodes can be of various types depending on the particular problem. In the context of SR, the node can either be a terminal, i.e., a constant or a variable, or some operator or function chosen from a set of functions defined by the user for the problem at hand. The individuals are interconnected in the left-to-right manner, meaning that an individual can act as an input operand only of those individuals which are positioned to its right in the population. Thus, the whole population represents a graph structure with multiple expressions rooted in the individual nodes. Expressions rooted in the function nodes can represent nonlinear symbolic functions of various complexity. The population is evolved through a local search procedure using a single reversible mutation operator.

MGGP is a tree-based genetic programming algorithm utilizing multiple linear regression. The main idea behind MGGP is that each individual is composed of multiple independent expression trees, called genes, which are put together by a linear combination to form a single final expression. The parameters of this top-level linear combination are computed using multiple linear regression where each gene acts as an independent feature. In this chapter, we build upon a particular implementation of MGGP – GPTIPS2 [127]. This particular instance of MGGP uses two crossover operators: (i) high-level crossover that combines gene sets of two parents; (ii) low-level crossover which is a classical Koza-style [84] subtree crossover operating on corresponding pairs of parental genes. Also, there are two mutation operators: (i) subtree mutation, which is a classical Koza-style subtree mutation; (ii) constant mutation, which alters the numerical values of leaves representing constants. Both the crossover and mutation operators are chosen stochastically.

Detailed explanation of these algorithms and their parameters is beyond the scope of this work and we refer the interested reader to [87] and [145].

3.3.3. COMPUTATIONAL COMPLEXITY

The computational complexity of the symbolic regression algorithms used in this work increases linearly with the size of the training data set as well as with the dimensionality of the problem. For example, considering a problem with one-dimensional state, one-dimensional input, one-dimensional output, and a data set of 1000 samples, a single run of the SNGP or MGGP algorithm with the default configuration takes about 3 minutes on a single core of a standard desktop PC. For a system with a 14-dimensional regressor and a 6-dimensional input, a single run takes up to 20 minutes.

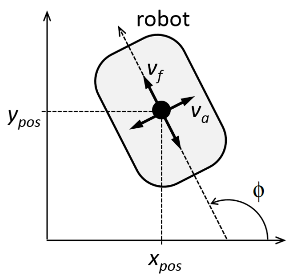
3.4. Experimental Results

We have carried out experiments with three nonlinear systems: a mobile robot, a 1-DOF inverted pendulum and a bipedal walking robot. The data, the codes and the detailed configuration of the experiments is available in our repository[1].

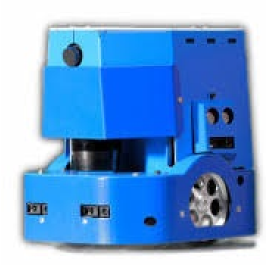
The simulation experiment with the mobile robot illustrates the use of the presented method, showing the precision and compactness of the models found in the case where the ground truth is known (Section 3.4.1). We show that the method is not dependent on the particular choice of the SR algorithm by comparing the performance of two SR methods, SNGP and MGGP. The subsequent experiment with the walking robot presents a more complex example and shows the performance of the method in a high-dimensional space (Section 3.4.2). With this example, we demonstrate the ability of the method to construct standard state-space models as well as input–output (NARX) models and we show how the method performs compared to two deep neural networks with different architectures. We conclude our set of experiments with the inverted pendulum system (Section 3.4.3). Similarly as in the experiment with the mobile robot, we evaluate the method with SNGP and MGGP, and we compare the results to two alternative approaches: neural networks and local linear regression. In addition to measuring the model prediction error, we perform real-time closed-loop control experiments with a lab setup to evaluate the performance of the algorithm in real-world conditions.

3.4.1. Mobile Robot

The state of a two-wheel mobile robot, see Figure 3.2 and [52], is described by $\mathbf{x} = (x_{pos}, y_{pos}, \phi)^\top$, with x_{pos} and y_{pos} the position coordinates and ϕ the heading. The control input is $\mathbf{u} = (v_f, v_a)^\top$, where v_f represents the forward velocity and v_a the angular velocity of the robot.



(a)



(b)

Figure 3.2: Mobile robot schematic (a) and photograph (b).

The continuous-time dynamic model of the robot is:

$$\begin{aligned} \dot{x}_{pos} &= v_f \cos(\phi), \\ \dot{y}_{pos} &= v_f \sin(\phi), \\ \dot{\phi} &= v_a. \end{aligned} \tag{3.7}$$

[1]https://github.com/erik-derner/symbolic-regression

DATA SETS

We generated a noise-free data set by using the Euler method to simulate the differential equations (3.7). With a sampling period $T_s = 0.05\,\text{s}$, the discrete-time approximation of (3.7) becomes:

$$\begin{aligned} x_{pos,k+1} &= x_{pos,k} + 0.05\,v_{f,k}\cos(\phi), \\ y_{pos,k+1} &= y_{pos,k} + 0.05\,v_{f,k}\sin(\phi), \\ \phi_{k+1} &= \phi_k + 0.05\,v_{a,k}. \end{aligned} \tag{3.8}$$

We generated training data sets of different sizes n_s. The initial state x_0 and the control input u_k for the whole simulation were randomly chosen from the ranges:

$$\begin{aligned} x_{pos} &\in [-1,1]\,\text{m}, \\ y_{pos} &\in [-1,1]\,\text{m}, \\ \phi &\in [-\pi,\pi]\,\text{rad}, \\ v_f &\in [-1,1]\,\text{m}\cdot\text{s}^{-1}, \\ v_a &\in \left[-\frac{\pi}{2},\frac{\pi}{2}\right]\text{rad}\cdot\text{s}^{-1}. \end{aligned} \tag{3.9}$$

A test data set was generated in order to assess the quality of the analytic models on data different from the training set. The test data set entries were sampled on a regular grid with 11 points spanning evenly each state and action component domain, as defined by (3.9). These samples were stored together with the next states calculated by using the Euler approximation.

EXPERIMENT SETUP

The purpose of this experiment was to test the ability of the SNGP and MGGP algorithms to recover from the data the analytic process model described by a known state-transition function. In order to assess the performance depending on the size of the data set and the complexity of the model, different combinations of the number of features n_f and the size of the training set n_s were tested. As the used algorithms only allow modeling one output at a time, they were run independently for each of the state components $x_{pos,k+1}$, $y_{pos,k+1}$ and ϕ_{k+1}.

The size of the SNGP population was set to 500 individuals and the evolution duration to 30000 generations. The set of elementary functions was defined as $\mathscr{F} = \{*, +, -, \sin, \cos\}$. The maximum depth d of the evolved nonlinear functions was set to 7 and the number of features was $n_f \in \{1,2,10\}$. To ensure a fair evaluation, the parameters of the MGGP algorithm were set similarly to provide both methods with a comparable amount of computational resoruces, taking into account the conceptual differences between the two algorithms.

RESULTS

The models found by symbolic regression were evaluated by calculating the RMSE median on the test data set over 30 independent runs of the SR algorithm. Note that each run yields a different model because the evolution process is guided by a unique sequence of random numbers. The results are listed in Table 3.1.

Table 3.1: Comparison of analytic process models for the experiment with the mobile robot. The table shows the RMSE medians over 30 runs of the SNGP (grey) and MGGP (white) algorithm for different numbers of features n_f and different numbers of training samples n_s.

Variable	n_f	Number of training samples n_s					
		20	50	100	200	500	1000
x_{pos}	1	1.77×10^{-1}	1.37×10^{-1}	9.98×10^{-2}	7.88×10^{-1}	3.25×10^{-2}	2.87×10^{-2}
		9.03×10^{-1}	7.66×10^{-1}	6.97×10^{-1}	9.16×10^{-1}	3.88×10^{-2}	3.29×10^{-2}
	2	6.44×10^{-8}	3.19×10^{-9}	2.05×10^{-9}	3.55×10^{-9}	5.93×10^{-9}	1.53×10^{-9}
		2.21×10^{-9}	5.64×10^{-10}	5.15×10^{-6}	1.11×10^{-2}	1.30×10^{-10}	1.37×10^{-10}
	10	3.78×10^{-5}	2.29×10^{-7}	1.09×10^{-7}	1.45×10^{-7}	5.21×10^{-9}	2.68×10^{-9}
		2.37×10^{-5}	1.39×10^{-7}	8.50×10^{-9}	5.54×10^{-9}	2.70×10^{-9}	5.26×10^{-10}
y_{pos}	1	9.06×10^{-1}	4.34×10^{-1}	1.39×10^{-1}	1.74×10^{-1}	3.21×10^{-2}	3.16×10^{-2}
		8.74×10^{-1}	9.47×10^{-1}	7.75×10^{-1}	7.33×10^{-1}	3.31×10^{-2}	3.16×10^{-2}
	2	4.87×10^{-1}	1.81×10^{-8}	1.18×10^{-8}	4.09×10^{-9}	1.93×10^{-8}	2.39×10^{-8}
		3.39×10^{-1}	3.38×10^{-2}	2.89×10^{-10}	2.76×10^{-10}	2.68×10^{-10}	2.14×10^{-2}
	10	4.48×10^{-4}	2.04×10^{-7}	4.11×10^{-7}	1.91×10^{-7}	1.60×10^{-8}	1.25×10^{-8}
		9.32×10^{-5}	1.16×10^{-7}	7.54×10^{-9}	6.45×10^{-9}	2.34×10^{-9}	8.33×10^{-10}
ϕ	1	9.81×10^{-2}	2.60×10^{-2}	6.44×10^{-4}	6.57×10^{-5}	6.79×10^{-4}	5.55×10^{-3}
		3.38×10^{0}	3.19×10^{0}	2.49×10^{-2}	1.34×10^{-3}	5.51×10^{-5}	5.48×10^{-5}
	2	7.05×10^{-8}	1.36×10^{-8}	6.16×10^{-9}	3.78×10^{-8}	7.78×10^{-9}	5.16×10^{-8}
		1.47×10^{-9}	5.08×10^{-10}	4.16×10^{-10}	4.02×10^{-10}	4.00×10^{-10}	4.01×10^{-10}
	10	5.35×10^{-6}	1.85×10^{-6}	2.09×10^{-6}	4.01×10^{-8}	6.00×10^{-9}	3.69×10^{-8}
		6.45×10^{-8}	9.34×10^{-9}	2.87×10^{-9}	1.35×10^{-9}	4.07×10^{-10}	4.00×10^{-10}

An example of a process model found by running SNGP with the parameters $n_f = 2$ and $n_s = 100$ is:

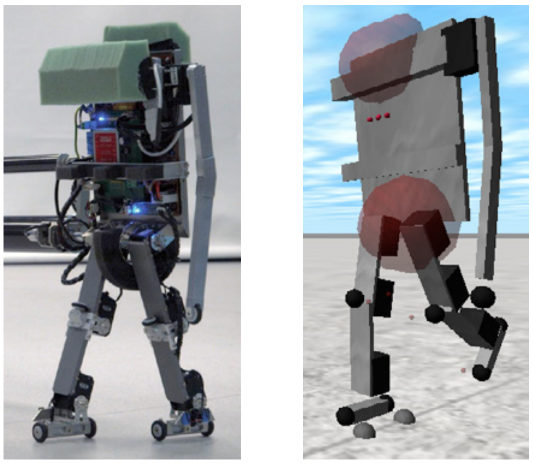
$$
\begin{aligned}
\hat{x}_{pos,k+1} &= 1.0\,x_{pos,k} + 0.0499998879\,v_{f,k}\cos(\phi_k)\,,\\
\hat{y}_{pos,k+1} &= 1.000000023\,y_{pos,k} + 0.0500000056\,v_{f,k}\sin(\phi_k) + 0.0000000191\,,\\
\hat{\phi}_{k+1} &= 0.9999982931\,\phi_k + 0.0500000536\,v_{a,k} - 0.0000059844\,.
\end{aligned}
\tag{3.10}
$$

The coefficients are rounded to 10 decimal digits in order to demonstrate the magnitude of the error compared to the original Euler approximation (3.8). The results show that even with a small training data set, a precise, parsimonious analytic process model can be found based on noise-free data.

The results also demonstrate how the number of features n_f plays an important role in the setting of the experiment parameters. In general, the RMSE decreases with an increasing number of features, whereas the complexity naturally grows by adding more features to the final model (3.5). The higher RMSE error when using only one feature is caused mainly by the fact that all parameters have to be evolved by the genetic algorithm, which is hard. On the other hand, when using more features, the least squares method can quickly and accurately find the coefficients of the features. These results support our choice to define the class of analytic models as a linear combination of features, as explained in Section 3.3.1. As a corollary, if the outline of the model structure is known in advance, it is recommended to set the number of features at least equal to the number of terms expected in the underlying function. Otherwise, it is advisable to set the number of features large enough, e.g. $n_f = 10$.

3.4.2. WALKING ROBOT

The robot LEO is a 2D bipedal walking robot [126], see Figure 3.3. It has 7 actuators: two in the ankles, knees and hips and one in its shoulder that allows the robot to stand up after a fall. LEO is connected to a boom with a parallelogram construction. This keeps the hip axis always horizontal, which makes it effectively a 2D robot and thus eliminates the sideways stability problem.



(a) (b)

Figure 3.3: The walking robot LEO: photograph (a) and simulation model rendering (b). [83].

The state vector of LEO $\mathbf{x} = (\boldsymbol{\psi}, \dot{\boldsymbol{\psi}})^\top$ consists of 14 components, where

$$\boldsymbol{\psi} = (\psi_{TRS}, \psi_{LH}, \psi_{RH}, \psi_{LK}, \psi_{RK}, \psi_{LA}, \psi_{RA})^\top \tag{3.11}$$

represents the angles of the torso, left and right hip, the knee and the ankle. Likewise,

$$\dot{\boldsymbol{\psi}} = (\dot{\psi}_{TRS}, \dot{\psi}_{LH}, \dot{\psi}_{RH}, \dot{\psi}_{LK}, \dot{\psi}_{RK}, \dot{\psi}_{LA}, \dot{\psi}_{RA})^\top \tag{3.12}$$

are the angular velocities of the torso, hips, knees and ankles. The action space of LEO comprises the voltage inputs to the seven joint actuators.

DATA SETS

In order to apply symbolic regression, the walking robot LEO was modeled using the Rigid Body Dynamics Library [54] and the data sets were generated using the Generic Reinforcement Learning Library [30], which allowed us to record trajectories while the robot was learning to walk.

We split the data set into two disjoint subsets: a training set and a test set. Both subsets are composed of consecutive samples from the simulation, which was run with a sampling period $T_s = 0.03$ s.

EXPERIMENT SETUP

The experiment was designed to evaluate the performance of our method on a more complex, high-dimensional example and to construct input–output (NARX) models in addition to the standard state-space models. We chose to use only the SNGP algorithm for this experiment and the main parameters were configured as follows. The population size was set to 500 and the number of generations to 30000. The depth limit d was fixed to 5 and the number of features was $n_f \in \{1, 5, 10\}$. The elementary function set was $\mathscr{F} = \{*, +, -, \sin, \cos, \mathrm{square}, \mathrm{cube}\}$.

During the simulation used to obtain the data sets, the shoulder was not actuated. Therefore, the input vector had only 6 components, one for each actuator. As in the case of the mobile robot, SNGP was run separately for each of the 14 state components.

In *Experiment B1*, we used SR to generate standard state-space models. In *Experiment B2*, we generated input–output models with the regression vector defined as $\boldsymbol{\varphi} = (\psi_k, \psi_{k-1}, u_{k-1})^\top$.

RESULTS

In order to evaluate the ability of SNGP to approximate the state-transition function, we calculated the RMSE medians over 30 runs of the algorithm on the test data set. The results for the state-space models are reported in Table 3.2 and for the input–output models in Table 3.3.

The results show the expected trend, which can be seen in all experiments: the quality of the models improves with the size of the training data set. However, it is noteworthy that the difference between the RMSE for models trained on 100 samples and for those trained on 5000 samples are in most cases negligible. This confirms our earlier observation that SR can be used to find accurate analytic process models on batches of data as small as 100 samples [44] even for high-dimensional systems.

The results for the input–output models are generally just slightly worse than those for the state-space models, with the benefit of speeding up the algorithm by reducing the number of modeled variables to a half.

COMPARISON WITH ALTERNATIVE METHODS

Deep neural networks are widely used to model an unknown system. In order to compare our method to alternative state-of-the-art methods, we have constructed two different neural networks:

- Deep neural network DNN-A was implemented in PyTorch. It consists of an input linear layer of size 20×200, followed by three linear layers with the size of 200×200, with a ReLU activation function used after each linear layer. The output layer has 200×14 units. The batch size was set to 32. The SGD algorithm [78] was used with the learning rate of 8.5×10^{-4}.

- Deep neural network DNN-B was implemented in TensorFlow. It is a fully connected network with 1 hidden layer, consisting of 512 units with ELU nonlinearity and 50% dropout. The batch size was set to 8. The Adam optimizer [79] was used with a learning rate of 10^{-3} and early stopping.

Table 3.2: Comparison of the state-space analytic process models for the walking robot LEO in *Experiment B1*. The table shows the RMSE medians over 30 runs of the SNGP algorithm for varying number of features n_f and number of training samples n_s.

Variable	n_f	Number of training samples n_s					
		100	200	500	1000	2000	5000
ψ_{TRS}	1	4.09×10^{-2}	1.72×10^{-2}	4.15×10^{-2}	5.38×10^{-2}	5.46×10^{-2}	5.68×10^{-2}
	5	1.90×10^{-2}	1.55×10^{-2}	1.46×10^{-2}	1.40×10^{-2}	1.39×10^{-2}	1.38×10^{-2}
	10	2.01×10^{-2}	1.62×10^{-2}	1.44×10^{-2}	1.32×10^{-2}	1.29×10^{-2}	1.25×10^{-2}
ψ_{LH}	1	2.99×10^{-2}	2.99×10^{-2}	3.60×10^{-2}	2.24×10^{-2}	2.34×10^{-2}	8.81×10^{-2}
	5	2.78×10^{-2}	2.38×10^{-2}	2.15×10^{-2}	2.07×10^{-2}	2.02×10^{-2}	2.01×10^{-2}
	10	2.99×10^{-2}	2.53×10^{-2}	2.20×10^{-2}	2.06×10^{-2}	1.95×10^{-2}	1.92×10^{-2}
ψ_{RH}	1	1.05×10^{-1}	9.16×10^{-2}	4.01×10^{-2}	3.71×10^{-2}	3.36×10^{-2}	2.84×10^{-2}
	5	3.81×10^{-2}	3.19×10^{-2}	2.69×10^{-2}	2.71×10^{-2}	2.61×10^{-2}	2.56×10^{-2}
	10	4.15×10^{-2}	3.54×10^{-2}	2.65×10^{-2}	2.65×10^{-2}	2.46×10^{-2}	2.46×10^{-2}
ψ_{LK}	1	5.52×10^{-2}	8.01×10^{-2}	2.43×10^{-2}	2.35×10^{-2}	2.40×10^{-2}	2.28×10^{-2}
	5	3.10×10^{-2}	2.68×10^{-2}	2.29×10^{-2}	2.15×10^{-2}	2.06×10^{-2}	2.07×10^{-2}
	10	3.43×10^{-2}	2.87×10^{-2}	2.22×10^{-2}	2.10×10^{-2}	1.97×10^{-2}	1.89×10^{-2}
ψ_{RK}	1	2.82×10^{-2}	2.36×10^{-2}	2.31×10^{-2}	2.31×10^{-2}	2.15×10^{-2}	2.13×10^{-2}
	5	2.77×10^{-2}	2.28×10^{-2}	2.01×10^{-2}	2.01×10^{-2}	2.01×10^{-2}	1.90×10^{-2}
	10	3.08×10^{-2}	2.45×10^{-2}	1.96×10^{-2}	1.87×10^{-2}	1.86×10^{-2}	1.77×10^{-2}
ψ_{LA}	1	9.31×10^{-2}	1.11×10^{-1}	1.10×10^{-1}	1.10×10^{-1}	7.07×10^{-2}	5.74×10^{-2}
	5	5.18×10^{-2}	4.00×10^{-2}	3.11×10^{-2}	2.95×10^{-2}	2.80×10^{-2}	2.81×10^{-2}
	10	5.66×10^{-2}	4.31×10^{-2}	3.16×10^{-2}	2.92×10^{-2}	2.73×10^{-2}	2.62×10^{-2}
ψ_{RA}	1	4.65×10^{-2}	4.51×10^{-2}	4.24×10^{-2}	4.54×10^{-2}	4.33×10^{-2}	7.37×10^{-2}
	5	4.98×10^{-2}	4.49×10^{-2}	3.77×10^{-2}	3.66×10^{-2}	3.65×10^{-2}	3.52×10^{-2}
	10	5.35×10^{-2}	4.70×10^{-2}	3.84×10^{-2}	3.65×10^{-2}	3.50×10^{-2}	3.39×10^{-2}
$\dot{\psi}_{TRS}$	1	8.91×10^{-1}	8.51×10^{-1}	8.19×10^{-1}	7.99×10^{-1}	7.94×10^{-1}	7.84×10^{-1}
	5	1.07×10^{0}	8.72×10^{-1}	7.78×10^{-1}	7.19×10^{-1}	6.92×10^{-1}	6.86×10^{-1}
	10	1.20×10^{0}	9.27×10^{-1}	7.93×10^{-1}	7.00×10^{-1}	6.67×10^{-1}	6.41×10^{-1}
$\dot{\psi}_{LH}$	1	1.44×10^{0}	1.23×10^{0}	1.16×10^{0}	1.15×10^{0}	1.14×10^{0}	1.14×10^{0}
	5	2.22×10^{0}	1.41×10^{0}	1.17×10^{0}	1.15×10^{0}	1.11×10^{0}	1.08×10^{0}
	10	2.07×10^{0}	1.48×10^{0}	1.20×10^{0}	1.16×10^{0}	1.10×10^{0}	1.06×10^{0}
$\dot{\psi}_{RH}$	1	1.49×10^{0}	1.32×10^{0}	1.31×10^{0}	1.28×10^{0}	1.25×10^{0}	1.24×10^{0}
	5	1.92×10^{0}	1.47×10^{0}	1.38×10^{0}	1.25×10^{0}	1.17×10^{0}	1.14×10^{0}
	10	1.97×10^{0}	1.57×10^{0}	1.52×10^{0}	1.27×10^{0}	1.17×10^{0}	1.12×10^{0}
$\dot{\psi}_{LK}$	1	1.59×10^{0}	1.25×10^{0}	1.14×10^{0}	1.11×10^{0}	1.10×10^{0}	1.09×10^{0}
	5	1.79×10^{0}	1.47×10^{0}	1.15×10^{0}	1.09×10^{0}	1.05×10^{0}	9.94×10^{-1}
	10	1.90×10^{0}	1.57×10^{0}	1.20×10^{0}	1.11×10^{0}	1.08×10^{0}	9.87×10^{-1}
$\dot{\psi}_{RK}$	1	1.02×10^{0}	9.35×10^{-1}	9.18×10^{-1}	9.24×10^{-1}	9.16×10^{-1}	9.05×10^{-1}
	5	1.13×10^{0}	9.98×10^{-1}	9.40×10^{-1}	9.29×10^{-1}	8.64×10^{-1}	8.32×10^{-1}
	10	1.24×10^{0}	1.07×10^{0}	9.83×10^{-1}	9.63×10^{-1}	8.73×10^{-1}	8.20×10^{-1}
$\dot{\psi}_{LA}$	1	1.76×10^{0}	1.52×10^{0}	1.32×10^{0}	1.31×10^{0}	1.28×10^{0}	1.26×10^{0}
	5	2.01×10^{0}	1.63×10^{0}	1.29×10^{0}	1.24×10^{0}	1.14×10^{0}	1.10×10^{0}
	10	2.18×10^{0}	1.67×10^{0}	1.35×10^{0}	1.25×10^{0}	1.15×10^{0}	1.10×10^{0}
$\dot{\psi}_{RA}$	1	1.69×10^{0}	1.64×10^{0}	1.60×10^{0}	1.58×10^{0}	1.58×10^{0}	1.58×10^{0}
	5	1.84×10^{0}	1.75×10^{0}	1.52×10^{0}	1.48×10^{0}	1.43×10^{0}	1.38×10^{0}
	10	1.92×10^{0}	1.86×10^{0}	1.62×10^{0}	1.51×10^{0}	1.43×10^{0}	1.35×10^{0}

Table 3.3: Comparison of the input–output analytic process models for the walking robot LEO in *Experiment B2*. The table shows the RMSE medians over 30 runs of the SNGP algorithm for varying number of features n_f and number of training samples n_s.

Variable	n_f	Number of training samples n_s					
		100	200	500	1000	2000	5000
ψ_{TRS}	1	5.78×10^{-2}	5.77×10^{-2}	5.71×10^{-2}	5.58×10^{-2}	5.60×10^{-2}	5.69×10^{-2}
	5	3.84×10^{-2}	3.60×10^{-2}	3.15×10^{-2}	2.65×10^{-2}	2.45×10^{-2}	2.33×10^{-2}
	10	4.07×10^{-2}	3.36×10^{-2}	2.88×10^{-2}	2.48×10^{-2}	2.09×10^{-2}	2.06×10^{-2}
ψ_{LH}	1	6.75×10^{-2}	6.57×10^{-2}	6.57×10^{-2}	5.90×10^{-2}	1.07×10^{-1}	6.67×10^{-2}
	5	3.80×10^{-2}	2.97×10^{-2}	2.62×10^{-2}	2.71×10^{-2}	2.56×10^{-2}	2.53×10^{-2}
	10	3.85×10^{-2}	3.15×10^{-2}	2.65×10^{-2}	2.64×10^{-2}	2.46×10^{-2}	2.40×10^{-2}
ψ_{RH}	1	8.04×10^{-2}	8.57×10^{-2}	6.62×10^{-2}	1.14×10^{-1}	1.06×10^{-1}	7.03×10^{-2}
	5	4.92×10^{-2}	3.81×10^{-2}	3.29×10^{-2}	3.20×10^{-2}	3.11×10^{-2}	3.04×10^{-2}
	10	5.40×10^{-2}	4.01×10^{-2}	3.25×10^{-2}	3.08×10^{-2}	2.88×10^{-2}	2.85×10^{-2}
ψ_{LK}	1	8.06×10^{-2}	5.52×10^{-2}	8.22×10^{-2}	7.52×10^{-2}	7.52×10^{-2}	5.77×10^{-2}
	5	3.66×10^{-2}	2.95×10^{-2}	2.46×10^{-2}	2.31×10^{-2}	2.20×10^{-2}	2.10×10^{-2}
	10	3.96×10^{-2}	3.09×10^{-2}	2.44×10^{-2}	2.21×10^{-2}	2.09×10^{-2}	2.04×10^{-2}
ψ_{RK}	1	8.69×10^{-2}	3.65×10^{-2}	2.99×10^{-2}	2.56×10^{-2}	8.88×10^{-2}	3.83×10^{-2}
	5	3.20×10^{-2}	2.62×10^{-2}	2.36×10^{-2}	2.26×10^{-2}	2.20×10^{-2}	2.18×10^{-2}
	10	3.49×10^{-2}	2.76×10^{-2}	2.24×10^{-2}	2.16×10^{-2}	2.08×10^{-2}	2.01×10^{-2}
ψ_{LA}	1	7.50×10^{-2}	1.07×10^{-1}	4.41×10^{-2}	1.03×10^{-1}	5.85×10^{-2}	1.03×10^{-1}
	5	5.44×10^{-2}	4.22×10^{-2}	3.27×10^{-2}	3.00×10^{-2}	2.89×10^{-2}	2.79×10^{-2}
	10	5.94×10^{-2}	4.62×10^{-2}	3.26×10^{-2}	2.96×10^{-2}	2.75×10^{-2}	2.66×10^{-2}
ψ_{RA}	1	1.19×10^{-1}	1.20×10^{-1}	9.65×10^{-2}	9.63×10^{-2}	1.02×10^{-1}	5.11×10^{-2}
	5	5.10×10^{-2}	4.45×10^{-2}	3.79×10^{-2}	3.69×10^{-2}	3.65×10^{-2}	3.60×10^{-2}
	10	5.51×10^{-2}	4.49×10^{-2}	3.79×10^{-2}	3.59×10^{-2}	3.44×10^{-2}	3.38×10^{-2}

We chose the RMSE medians of SNGP with $n_s = 1000$ and $n_f = 10$ as the benchmark configuration. State-space models were used in this scenario. The training and test sets were the same for all compared methods. Figure 3.4 shows an overview of the performance of the two variants of DNN compared to the SNGP algorithm and detailed results are presented in Table 3.4. The results show that the SNGP algorithm is able to find substantially better models than the neural networks for the angles, while the performance on the angular velocities is comparable among all the tested methods.

Table 3.4: Comparison of the RMSE of the state-space process models calculated on the test data set for the walking robot LEO using two variants of a deep neural network (DNN-A and DNN-B) and SNGP. The reference configuration of SNGP used for this comparison was $n_f = 10$ and $n_s = 1000$.

Variable	Method		
	DNN-A	DNN-B	SNGP
ψ_{TRS}	1.33×10^{-1}	9.27×10^{-2}	1.32×10^{-2}
ψ_{LH}	1.86×10^{-1}	1.54×10^{-1}	2.06×10^{-2}
ψ_{RH}	2.08×10^{-1}	1.23×10^{-1}	2.65×10^{-2}
ψ_{LK}	2.24×10^{-1}	1.37×10^{-1}	2.10×10^{-2}
ψ_{RK}	2.02×10^{-1}	1.10×10^{-1}	1.87×10^{-2}
ψ_{LA}	1.62×10^{-1}	1.24×10^{-1}	2.92×10^{-2}
ψ_{RA}	1.54×10^{-1}	9.36×10^{-2}	3.65×10^{-2}
$\dot{\psi}_{TRS}$	7.39×10^{-1}	6.38×10^{-1}	7.00×10^{-1}
$\dot{\psi}_{LH}$	1.13×10^{0}	1.12×10^{0}	1.16×10^{0}
$\dot{\psi}_{RH}$	1.22×10^{0}	1.20×10^{0}	1.27×10^{0}
$\dot{\psi}_{LK}$	1.08×10^{0}	1.06×10^{0}	1.11×10^{0}
$\dot{\psi}_{RK}$	9.49×10^{-1}	8.68×10^{-1}	9.63×10^{-1}
$\dot{\psi}_{LA}$	1.23×10^{0}	1.23×10^{0}	1.25×10^{0}
$\dot{\psi}_{RA}$	1.54×10^{0}	1.42×10^{0}	1.51×10^{0}

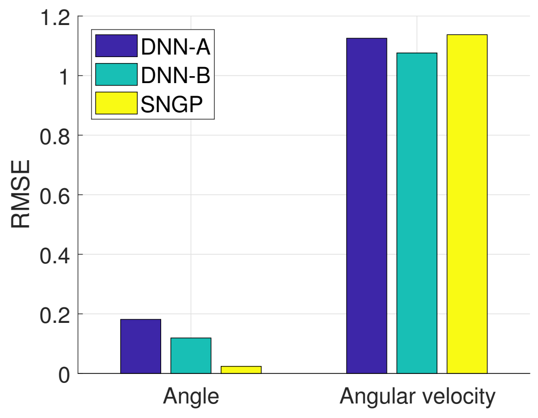


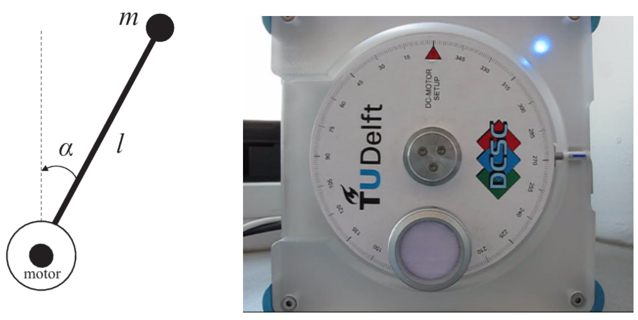
Figure 3.4: Comparison of two DNN variants with the SNGP algorithm on the walking robot example. The bars show the mean RMSE over the 7 angles and over the 7 angular velocities on the test data set.

3.4.3. INVERTED PENDULUM

The inverted pendulum system consists of a weight of mass m attached to an actuated link which rotates in the vertical plane, see Figure 3.5a. The state vector is $\mathbf{x} = (\alpha, \dot{\alpha})^\top$, where α is the angle and $\dot{\alpha}$ is the angular velocity of the link. The control input is the voltage u. The continuous-time model of the pendulum dynamics is:

$$\ddot{\alpha} = \frac{1}{J} \cdot \left(\frac{K}{R} u - m g l \sin(\alpha) - b \dot{\alpha} - \frac{K^2}{R} \dot{\alpha} - c \operatorname{sign}(\dot{\alpha}) \right) \tag{3.13}$$

with $J = 1.7937 \times 10^{-4}\,\mathrm{kg \cdot m^2}$, $m = 0.055\,\mathrm{kg}$, $g = 9.81\,\mathrm{m \cdot s^{-2}}$, $l = 0.042\,\mathrm{m}$, $K = 0.0536\,\mathrm{N \cdot m \cdot A^{-1}}$, $b = 1.94 \times 10^{-5}\,\mathrm{N \cdot m \cdot s \cdot rad^{-1}}$, $R = 9.5\,\Omega$ and $c = 8.5 \times 10^{-4}\,\mathrm{kg \cdot m^2 \cdot s^{-2}}$. The angle is $\alpha = 0\,\mathrm{rad}$ or $\alpha = 2\pi\,\mathrm{rad}$ for the pendulum pointing down and $\alpha = \pi$ for the pendulum pointing up.



(a) (b)

Figure 3.5: Inverted pendulum schematic (a) and the real inverted pendulum system (b).

The reward function used in the RL experiments was defined as follows:

$$\rho(x_k, u_k, x_{k+1}) = -0.5|\alpha_r - \alpha_k| - 0.01|\dot{\alpha}_r - \dot{\alpha}_k| - 0.05|u_k|, \tag{3.14}$$

where $(\alpha_r, \dot{\alpha}_r)^\top$ is a constant reference (goal) state.

DATA COLLECTION

As we will present an experiment with the inverted pendulum performing a control task, we start this section by a short overview of the data collection methods used. Two different situations can be distinguished: initial model learning and model learning under a given policy.

Initial Model Learning. At the beginning, when the control policy is not yet available, the system can be excited by a test signal in order to obtain a sufficiently rich data set. Various methods for designing suitable test signals are described in the literature, such as the generalized binary noise sequence [139]. The important parameters to be selected are the input signal amplitude, the way the random signal is generated (e.g., the 'switching' probability) and the experiment duration.

Model Learning Under a Given Policy. Once an acceptable control policy has been learned, the system can be controlled to execute the required task. Data can be collected while performing the control task and used to further improve the model. As the information captured in the data under steady operating conditions might not be sufficient in certain situations, the control input can be adjusted by adding a test signal in this case as well. The characteristics of this test signal are usually different from the one used for initial model learning; for instance, it typically has a lower amplitude.

DATA SETS

We used both simulated and real measured data in the experiments with the inverted pendulum. In all experiments, the discrete-time sampling period used was $T_s = 0.05\,\mathrm{s}$.

At first, we generated a noise-free data set for *Experiment C1* by using the Euler method to simulate the differential equation (3.13):

$$\begin{aligned} \alpha_{k+1} &= \alpha_k + 0.05\,\dot\alpha_k, \\ \dot\alpha_{k+1} &= 0.9102924564\,\dot\alpha_k - 0.2369404025\,\mathrm{sign}(\dot\alpha_k) \\ &\quad + 1.5727561084\,u_k - 6.3168590065\,\sin(\alpha_k). \end{aligned} \tag{3.15}$$

The data set for *Experiment C2* was created by integrating (3.13) by using the fourth-order Runge-Kutta method and adding Gaussian noise. The transformation from the original states $\mathbf{x} = (\alpha, \dot\alpha)^\top$ to the states with Gaussian noise $\mathbf{x}_n = (\alpha_n, \dot\alpha_n)^\top$ is defined as

$$\begin{aligned} \alpha_n &= \alpha + \pi\lambda r_{n,1}\,, \\ \dot\alpha_n &= \dot\alpha + 40\lambda r_{n,2}\,, \end{aligned} \tag{3.16}$$

where $r_{n,1}$, $r_{n,2}$ are random numbers drawn from a normal distribution with zero mean and a standard deviation of 1. The constant $\lambda \in \{0, 0.01, 0.05, 0.1\}$ controls the amount of noise and the constants π and 40 make sure that the added noise is approximately proportional to the range of each variable.

In both *Experiments C1* and *C2*, the initial state was $\alpha = 0$, $\dot\alpha = 0$ and the control input was chosen randomly at each time step k from the range $u_k \in [-5, 5]\,\mathrm{V}$.

The test data sets were created similarly as in Section 3.4.1. The samples were generated on a regular grid of $31 \times 31 \times 31$ points, spanning the state and action domain: $\alpha \in [-\pi, \pi]\,\mathrm{rad}$, $\dot\alpha \in [-40, 40]\,\mathrm{rad}\cdot\mathrm{s}^{-1}$ and $u \in [-5, 5]\,\mathrm{V}$. For all samples, the next states in the test set for *Experiment C1* were calculated using the Euler approximation. In *Experiment C2*, we generated a noise-free test set by applying the fourth-order Runge-Kutta method to all samples on the grid.

The real data for *Experiment C3* were measured on the real inverted pendulum system shown in Figure 3.5b. At first, the system was excited by applying a uniformly distributed random control input u_k within the range $[-5, 5]\,\mathrm{V}$ at each time step k. The random interaction with the system lasted for 5 seconds and the recorded data set comprised 100 samples. The data are shown in Figure 3.6. The data set was later enriched by samples recorded while applying the control policy (2.8) to perform the swing-up task on the real system, which will be described in the following section.

The sequences recorded for *Experiment C3* were split into training and test subsets. Every third sample was used for the test set, while the remaining samples formed the training set. In all experiments, the reported RMSE values were calculated on the respective test data set.

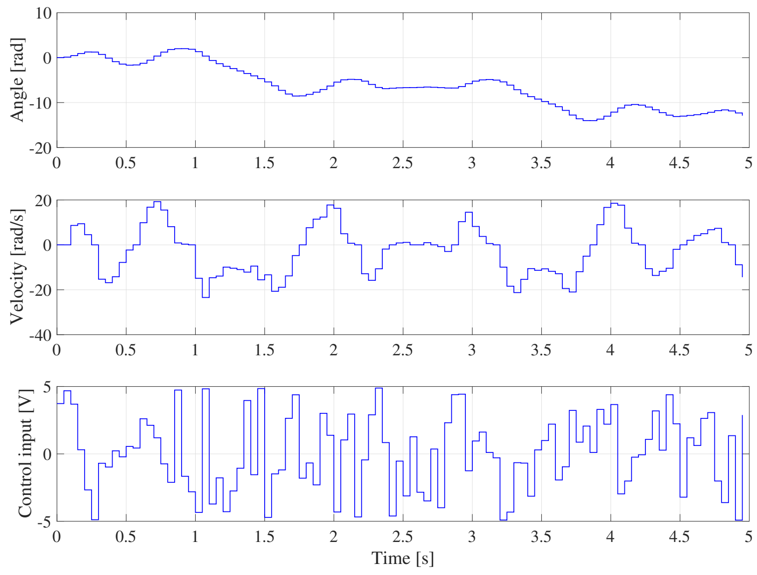


Figure 3.6: Initial data set obtained on the real inverted pendulum system as a response to the random input shown in the bottom panel.

Experiment Setup

Similarly as in the experiment with the mobile robot, the SNGP and MGGP algorithms were first employed in *Experiment C1* to test the ability of SR to generate precise models for the inverted pendulum system using a data set generated by the Euler method. The experiment serves to evaluate how the training data set size n_s and the number of features n_f influence the quality of the model.

Experiment C2 demonstrates how the analytic process models are evolved using the Runge-Kutta simulation data set with noise. The maximum number of features n_f in the symbolic regression algorithms was set to 10 in order to facilitate the evolution of models capturing the more complex underlying function. This experiment tests the behavior of the method in environments with noisy measurements.

We conclude the experiments with *Experiment C3*, which shows the intended use of the method within RL on the example of the underactuated swing-up task, performed on a real inverted pendulum system. The control goal is to stabilize the pendulum in the unstable equilibrium $\mathbf{x}_r = (\alpha_r, \dot{\alpha}_r)^\top = (\pi, 0)^\top$. As the input is limited to the range $u \in [-2, 2]$ V, the available torque is insufficient to push the pendulum directly up from the majority of initial states, and therefore it has to be first swung back and forth to gather energy. At first, we constructed 30 analytic models using the data set recorded under random input and then selected the model with the lowest RMSE on the test set. This *initial model* was employed to calculate the policy for the swing-up task, see Section 2.5. To find an approximation of the optimal value function, we used the fuzzy V-iteration algorithm [26]. We applied the policy to the real system in four independent runs, starting at the initial state $\mathbf{x}_0 = (0, 0)^\top$. In addition, we performed other four swing-ups with exploration noise added to the control input. The exploration noise was normally distributed with the standard deviation ranging from 0.2 to 0.5 V. All eight sequences, each consisting of approximately 50 measurements, were added to the initial data set recorded under random input. Using this extended data set, 30 refined analytic process models were

Table 3.5: Comparison of analytic process models for the inverted pendulum system in *Experiment C1*. The table shows the RMSE medians over 30 runs of the SNGP (grey) and MGGP (white) algorithm for varying number of features n_f and varying number of training samples n_s.

Variable	n_f	Number of training samples n_s					
		20	50	100	200	500	1000
α	1	9.19×10^{-4}	3.80×10^{-4}	2.45×10^{-2}	2.28×10^{-3}	1.89×10^{-3}	2.38×10^{-3}
		5.51×10^{-1}	3.33×10^{-1}	4.46×10^{-1}	3.15×10^{-1}	2.44×10^{-1}	3.25×10^{-1}
	2	2.09×10^{-7}	2.39×10^{-7}	1.06×10^{-7}	1.82×10^{-9}	1.94×10^{-8}	4.60×10^{-9}
		3.94×10^{-10}	3.78×10^{-10}	3.77×10^{-10}	3.76×10^{-10}	3.76×10^{-10}	3.76×10^{-10}
	10	5.03×10^{-9}	4.44×10^{-7}	4.41×10^{-9}	1.35×10^{-9}	8.45×10^{-10}	4.56×10^{-10}
		4.29×10^{-10}	3.87×10^{-10}	3.87×10^{-10}	3.80×10^{-10}	3.77×10^{-10}	3.76×10^{-10}
$\dot{\alpha}$	1	7.97×10^{-1}	3.17×10^{-1}	2.51×10^{-1}	2.61×10^{-1}	2.34×10^{-1}	5.11×10^{-1}
		9.21×10^{-1}	3.64×10^{-1}	2.42×10^{-1}	1.52×10^{-1}	3.42×10^{-1}	2.14×10^{-1}
	4	1.12×10^{-6}	5.61×10^{-7}	1.19×10^{-6}	1.61×10^{-6}	8.17×10^{-7}	6.75×10^{-7}
		1.73×10^{-9}	1.64×10^{-9}	1.56×10^{-9}	1.55×10^{-9}	1.51×10^{-9}	1.50×10^{-9}
	10	5.16×10^{-7}	1.83×10^{-7}	2.64×10^{-7}	4.40×10^{-7}	5.15×10^{-7}	2.66×10^{-6}
		1.90×10^{-9}	1.66×10^{-9}	1.60×10^{-9}	1.56×10^{-9}	1.54×10^{-9}	1.53×10^{-9}

learned and the model with the lowest error on the test set was chosen as the final *refined model*. Like in *Experiment C2*, the number of features was set to $n_f = 10$ to facilitate modeling the more complex state-transition function.

In all experiments, the size of the SNGP population was set to 500 and the evolution was limited to 30000 generations. The elementary function set was $\mathscr{F} = \{*, +, -, \sin, \cos, \text{sign}\}$. The maximum depth d was set to 7. In *Experiment C1*, various numbers of features were tested: $n_f \in \{1, 2, 10\}$ for α and $n_f \in \{1, 4, 10\}$ for $\dot{\alpha}$. The parameters of the MGGP algorithm in *Experiment C1* and *C2* were set similarly, taking into account the conceptual differences between the two algorithms to allow for a fair comparison.

RESULTS

The results of *Experiment C1* are summarized in Table 3.5 for the SNGP and MGGP algorithm. Similarly as in the previous examples, the results indicate that the precision of the models increases with increasing number of features. The overall performance of both SR algorithms is comparable.

An example of an analytic process model found with the parameters $n_f = 2$ for α, $n_f = 4$ for $\dot{\alpha}$ and $n_s = 20$ is:

$$\begin{aligned}\hat{\alpha}_{k+1} &= \alpha_k + 0.05\,\dot{\alpha}_k - 0.0000000001\,,\\ \hat{\dot{\alpha}}_{k+1} &= 0.9102924745\,\dot{\alpha}_k - 0.2369403835\,\text{sign}(\dot{\alpha}_k) + 1.5727561072\,u_k\\ &\quad - 6.3168589936\,\sin(\alpha_k) + 0.0000000013\,.\end{aligned} \tag{3.17}$$

The error of the analytic model w.r.t. the Euler approximation (3.15) is very small. These results confirm that the proposed method can find precise models even on small data sets.

The results of *Experiment C2* presented in Table 3.6 show that the analytic models are able to approximate the state-transition function well even on data with a reasonable amount of

Table 3.6: Comparison of analytic process models for the inverted pendulum system in *Experiment C2*. The table shows the comparison of the RMSE medians over 30 runs of the SNGP (grey) and MGGP (white) algorithm depending on the Gaussian noise standard deviation coefficient λ and the number of training samples n_s.

Variable	λ	Number of training samples n_s		
		20	100	1000
α	0	9.58×10^{-2}	1.79×10^{-2}	6.11×10^{-3}
		8.13×10^{-2}	1.05×10^{-2}	1.36×10^{-2}
	0.01	3.95×10^{-1}	1.45×10^{-1}	2.80×10^{-2}
		3.96×10^{-1}	1.37×10^{-1}	2.85×10^{-2}
	0.05	1.15×10^{0}	4.89×10^{-1}	1.43×10^{-1}
		8.69×10^{-1}	5.01×10^{-1}	1.42×10^{-1}
	0.1	1.90×10^{0}	7.61×10^{-1}	3.54×10^{-1}
		2.26×10^{0}	8.22×10^{-1}	3.59×10^{-1}
$\dot{\alpha}$	0	4.56×10^{0}	7.65×10^{-1}	5.04×10^{-1}
		3.89×10^{0}	7.56×10^{-1}	5.38×10^{-1}
	0.01	4.22×10^{0}	2.28×10^{0}	8.13×10^{-1}
		4.71×10^{0}	2.75×10^{0}	8.18×10^{-1}
	0.05	7.89×10^{0}	6.07×10^{0}	3.14×10^{0}
		7.39×10^{0}	6.75×10^{0}	2.76×10^{0}
	0.1	1.26×10^{1}	9.61×10^{0}	6.65×10^{0}
		1.26×10^{1}	8.99×10^{0}	6.53×10^{0}

noise. The use of the Runge-Kutta method to generate data sets leads to substantially more complicated models than when using the data generated by using the Euler method. Again, the performance of the SNGP and MGGP algorithm is comparable.

In *Experiment C3*, we have shown that SR is able to find analytic process models using data collected on the real system. Already after a short (5 s) interaction under the random input, an analytic process model is found which enables RL to perform the swing-up, see Figure 3.7a. Performing the swing-up task allows to collect more data in important parts of the state space around the trajectory to the goal state. Figure 3.7b shows that the performance of the model further improves after adding data collected while performing the swing-up task with the initial model. Figure 3.8 compares the swing-up response with the initial and the refined model. The histogram in Figure 3.9 and a two-sample t-test with unpooled variance applied to the discounted return show that the performance improvement between the policy based on the initial and the refined analytic process model is statistically significant ($p = 2 \times 10^{-22}$). The RMSE medians over 30 runs of the SNGP algorithm were 1.70×10^{-2} for α and 6.03×10^{-1} for $\dot{\alpha}$ in case of the initial model and 1.16×10^{-2} for α and 3.35×10^{-1} for $\dot{\alpha}$ in case of the refined model.

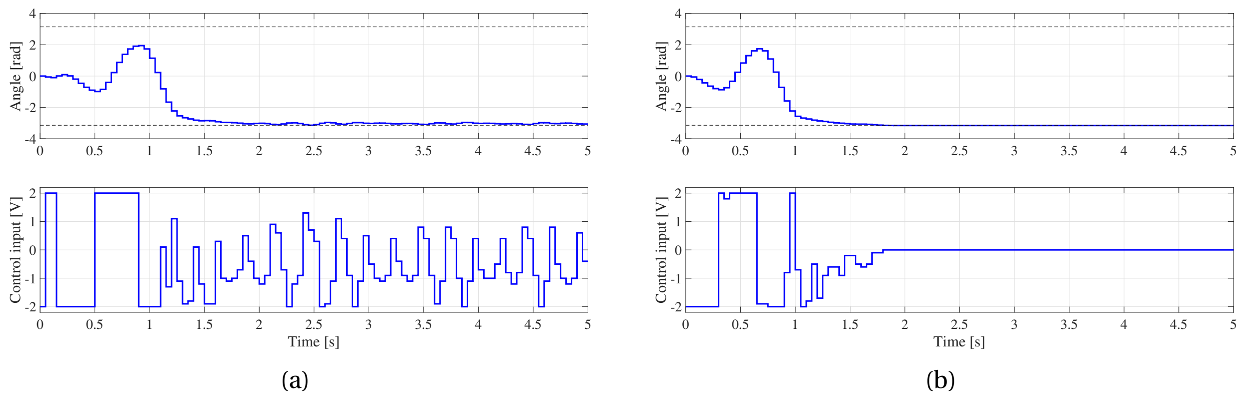


Figure 3.7: A typical real swing-up experiment with the initial model (a) and the refined model (b).

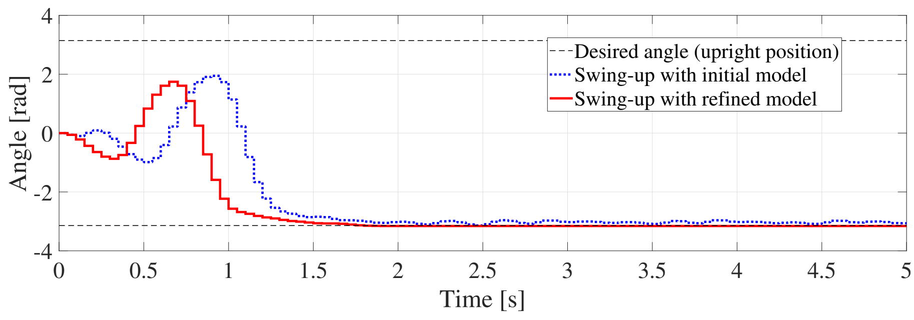


Figure 3.8: Comparison of the real swing-up response with the initial model, learned from the random data, and the refined model, learned from the random data merged with additional data from eight real swing-up experiments.

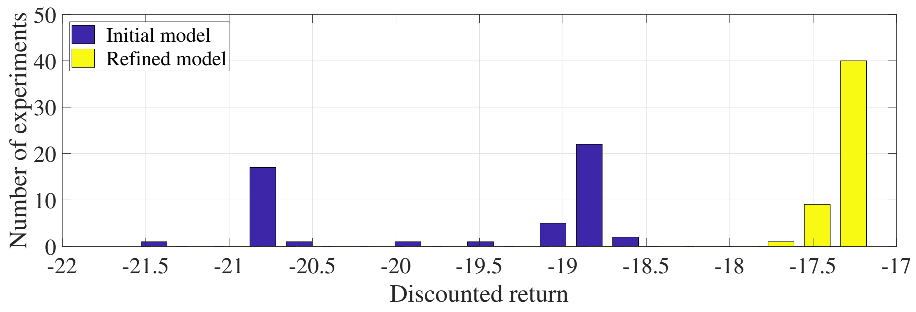


Figure 3.9: Histograms of 50 real experiments with the initial model and the refined model measured by the discounted return. The performance improvement is statistically significant ($p = 2 \times 10^{-22}$).

Comparison with Alternative Methods

We compared our modeling results with local linear regression (LLR) [10]. We selected the Runge-Kutta data set with 1000 samples and zero noise as a reference training set and the regular grid as a test set (see Section 3.4.3 for details). The LLR memory contained 1000 samples and the number of nearest neighbors was set to 10. The RMSE achieved by LLR was 1.73×10^{-1} for α and 6.93×10^{0} for $\dot{\alpha}$. In both cases, the SNGP algorithm achieved a better RMSE by at least one order of magnitude (6.11×10^{-3} for α and 5.04×10^{-1} for $\dot{\alpha}$).

We also compared the results of our method to a neural network. Given the relative simplicity of the problem, the network had one hidden layer, consisting of 40 neurons, and it was trained using the Levenberg-Marquardt algorithm. The number of neurons in the hidden layer was tuned by testing networks with 5 to 100 neurons and choosing the one that performed best on the test data. The RMSE achieved on the aforementioned reference data set was 6.82×10^{-2} for α and 2.59×10^{0} for $\dot{\alpha}$. Again, compared to the RMSE values achieved by our method (stated at the end of the previous paragraph), symbolic regression finds substantially better models in terms of RMSE compared to those found by the neural network.

3.5. Conclusions

We showed that symbolic regression is a very effective method for constructing dynamic process models from data. It generates parsimonious models in the form of analytic expressions, which makes it a good alternative to black-box models, especially in problems with limited amounts of data. Prior knowledge on the type of nonlinearities and model complexity can easily be included in the symbolic regression procedure. Despite the technique is not yet broadly used in the field of robotics and dynamic systems, we believe that it will become a standard tool for system identification.

The experiments with the walking robot demonstrate that symbolic regression can be used to construct precise process models even for high-dimensional systems. We have confirmed empirically that the computational complexity of the algorithm grows linearly with the dimensionality of the system. It is also worth mentioning that the complexity of the analytic models does not grow significantly with the complexity of the system.

The real-world experiment with the inverted pendulum shows that already after 5 seconds of interaction with the system, an initial analytic process model is found, which not only accurately predicts the process behavior, but also serves as a reliable model for the design of an RL controller. By collecting the data during several executions of the swing-up task using the initial analytic model and adding them to the data set used by SR to learn the model, the performance on the swing-up task further improves.

Our evaluation shows that two distinct symbolic regression algorithms, SNGP and MGGP, perform comparably well on the evaluated systems. This indicates that the proposed method is not dependent on the particular choice of the symbolic regression method. We compared the performance of symbolic regression with alternative state-of-the-art methods, in particular with neural networks and with local linear regression. The results show that the proposed method performs in most cases significantly better than the alternatives.

Another important outcome is that SR can be used to find both state-space and input–output models. The use of input–output models is beneficial because it does not require the observations of the full state vector and it also makes the algorithm faster because of modeling a reduced number of variables.

We have identified several possibilities for future extensions of this work. The main objective is to apply SR methods within the entire RL scheme, i.e., also for approximating the V-function, and also to use analytic models in combination with actor-critic online RL. In some cases, especially when using many features, analytic models tend to be unnecessarily complex. In our future work, we will investigate systematic reduction of analytic models.

4 Efficient Selection of Informative Samples for Model Learning

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 chapter, 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.

This chapter is an adapted version of the journal paper [47].

4.1. 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 [99, 112] 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 [38, 128], 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 [20]. 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 developed in the field of *active learning* [128], known mostly for applications in classification [51, 69, 150], but also applied to regression [28, 77, 149]. 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 [128]. 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 [20] and on the domain coverage [128], 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 [43]. 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 [38, 67, 107, 120]. As SR can be time-consuming for large data sets, selecting a suitable small training set makes it very well usable in practice.

This chapter 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 chapter is organized as follows. The model learning framework is described in Section 4.2. Sections 4.3 and 4.4 present the experimental results and Section 4.5 concludes the chapter.

4.2. METHODS

The definition of the nonlinear dynamic models and the theoretical background of SNGP is given in Chapter 2. The model learning procedure is explained in Section 4.2.1 and the sample-selection methods evaluated in this chapter are described in Section 4.2.2.

4.2.1. 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 2.4, 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 work, 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 Section 4.2.2.

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
```

4.2.2. 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.

Maximum Variance

A common state-of-the-art informed sample-selection method is based on the maximum variance between the model outputs [20, 41, 128]. 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.

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 [128] 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.

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 $f_r(\mathbf{x}_\ell, \mathbf{u}_\ell)$, $r = 1, 2, \dots, n_r$, 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:

$$\ell^* = \underset{\ell \in \{1,\dots,N\}}{\operatorname{argmax}} \frac{1}{n_r} \sum_{r=1}^{n_r} \left(f_r(\mathbf{x}_\ell, \mathbf{u}_\ell) - z_\ell \right)^2 , \tag{4.1}$$

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 (4.1) 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.

Sequential Addition

A common uninformed sample-selection approach is to build the model from samples added in the order as they are logged [112]. 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.

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 [128]. 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.

4.2.3. 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 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 4.3. 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 ($< 50\,\mathrm{ms}$) 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.

4.2.4. 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 work 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.

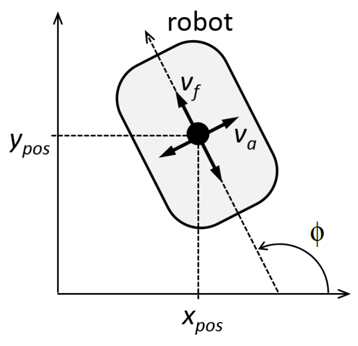
Finally, we assume that the buffer does not contain a large amount of corrupted data (outliers). We have empirically evaluated that the method is robust to a small number of erroneous records (less than 10 %) present in the training set.

4.3. MOBILE ROBOT EXPERIMENTS

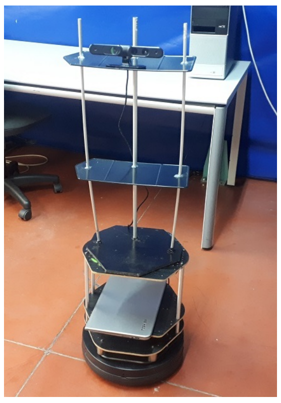
We have chosen a mobile robot as a suitable benchmark for our method. We have carried out experiments both in simulations and with a real mobile robot TurtleBot 2.

4.3.1. SYSTEM DESCRIPTION

We consider a two-wheeled mobile robot shown in Figure 4.1. 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 $\mathbf{u} = (v_f, v_a)^\top$.



(a)



(b)

Figure 4.1: Mobile robot: a) schematic, b) photo of the TurtleBot used in the experiments.

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

$$\begin{aligned} \dot{x}_{pos} &= v_f \cos(\phi), \\ \dot{y}_{pos} &= v_f \sin(\phi), \\ \dot{\phi} &= v_a, \end{aligned} \tag{4.2}$$

neglecting the dynamics caused by the robot's inertia and actuators. This model is only used for simulations in Section 4.3.5. In our experiments, the inputs are limited to the domain $v_f \in [0, 0.3]\ \mathrm{m \cdot s^{-1}}$ and $v_a \in [-1, 1]\ \mathrm{rad \cdot s^{-1}}$, which substantiates the use of the sampling period $T_s = 0.2$ s.

4.3.2. 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:

$$\mathbf{s}_k = (x_{pos,k}, y_{pos,k}, \phi_k, v_{f,k}, v_{a,k}, x_{pos,k+1}, y_{pos,k+1}, \phi_{k+1}), \tag{4.3}$$

where k denotes the time step, see (2.1). Odometry measurements were used to record the samples in the experiments with the real robot.

4.3.3. 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 $\mathscr{F} = \{\times, +, -, \sin, \cos, \text{square}, \text{cube}\}$.

We have evaluated the five sample-selection methods described in Section 4.2.2. 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.

4.3.4. 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 work; for details, please refer to [26]. We set the discount factor γ to 0.99. The reward function is equal to zero if the robot is within ±0.01 m in x_{pos}, ±0.01 m in y_{pos}, and ±0.02 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 $x_{pos} \in [0,1]$ m, $y_{pos} \in [0,1]$ m, and $\phi \in (-\pi, \pi]$ 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 $v_f \in [0, 0.2]\,\mathrm{m \cdot s^{-1}}$ and from 21 values spanning evenly the range $v_a \in [-0.5, 0.5]\,\mathrm{rad \cdot s^{-1}}$.

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 $(0.8, 0.2, 0)^\top$.

4.3.5. 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 [29] to the equations of motion (4.2). Note that the simulation model does not constitute a part of our method; it only serves to generate the data samples.

Table 4.1 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.

The results in Table 4.1 and in Figure 4.2 show that the informative sample-selection methods allow for constructing accurate 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 4.1 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 Figure 4.2. For instance, the median

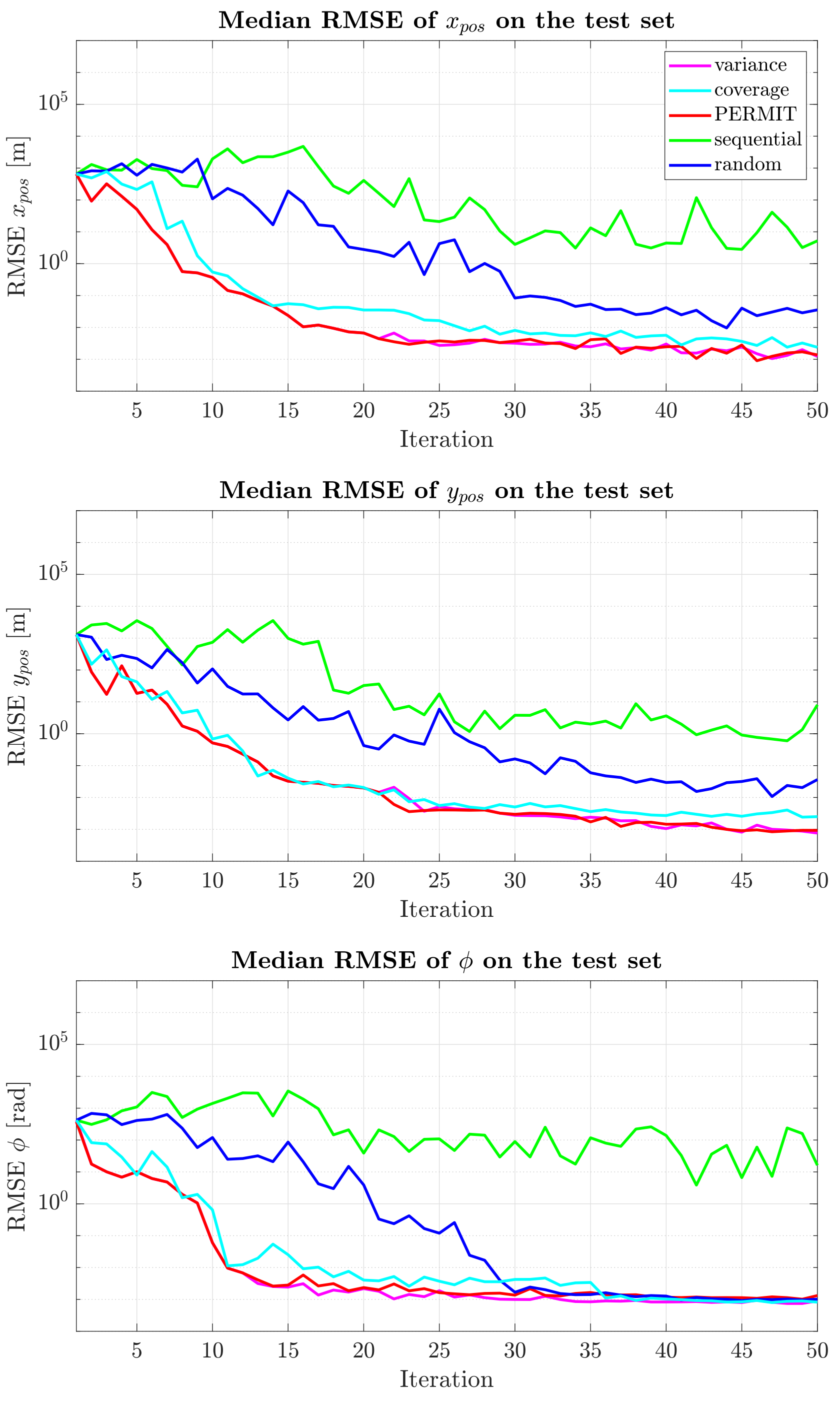


Figure 4.2: Evaluation of the sample-selection methods for modeling all variables in the experiment with the simulated mobile robot. The number of training samples starts at five in the first iteration and increases by one in each iteration.

Table 4.1: 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	2.52×10^1	3.16×10^1	9.58×10^0	4.13×10^1	3.82×10^0	3.21×10^1
	coverage	5.71×10^1	4.04×10^1	1.36×10^1	7.07×10^1	4.59×10^0	7.14×10^1
	PERMIT	2.52×10^1	3.16×10^1	8.68×10^0	4.09×10^1	3.86×10^0	3.51×10^1
	sequential	6.21×10^2	5.30×10^2	5.92×10^2	1.65×10^3	5.27×10^2	1.93×10^3
	random	2.02×10^2	8.14×10^1	8.37×10^1	1.76×10^2	6.73×10^1	1.25×10^2
Average RMSE spread	variance	9.18×10^5	9.32×10^{15}	6.38×10^5	3.38×10^{15}	2.02×10^{13}	5.51×10^{12}
	coverage	1.90×10^7	4.26×10^{21}	2.34×10^{27}	1.59×10^{11}	6.17×10^{10}	8.08×10^{33}
	PERMIT	9.18×10^5	9.32×10^{15}	6.27×10^5	3.38×10^{15}	2.02×10^{13}	5.51×10^{12}
	sequential	4.80×10^{35}	1.31×10^{28}	1.13×10^{30}	2.55×10^{33}	1.18×10^{58}	6.86×10^{30}
	random	7.59×10^{21}	3.71×10^{14}	1.58×10^{11}	8.33×10^{13}	2.53×10^{21}	6.70×10^{26}

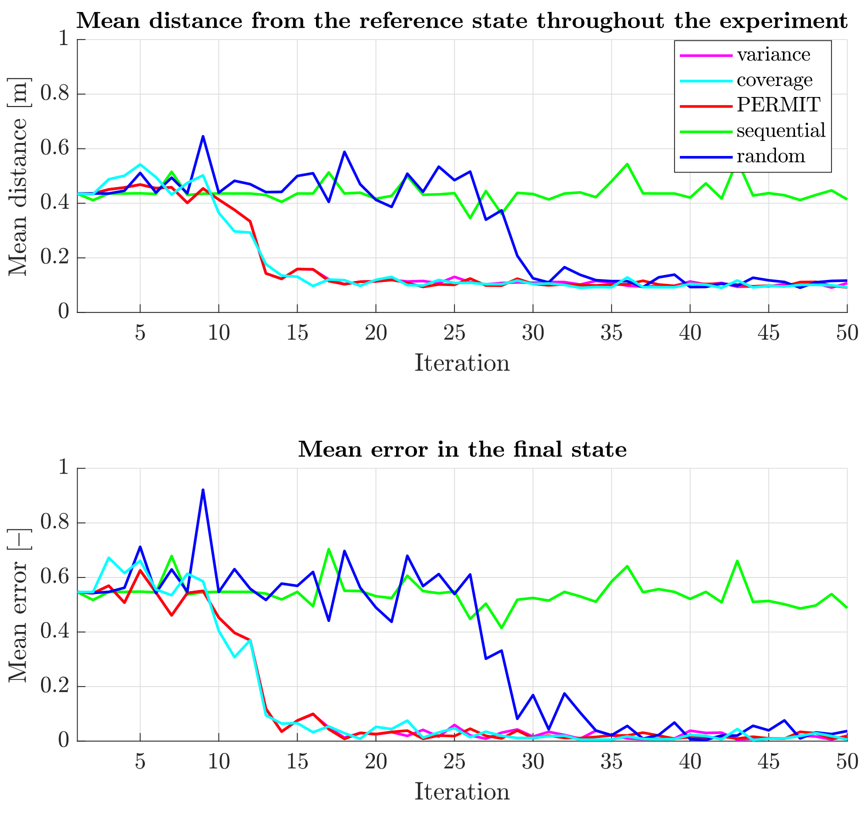


Figure 4.3: Evaluation of the control task executions performed with the simulated mobile robot.

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 Figure 4.3. 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.

4.3.6. Results with the Real Mobile Robot

We have performed lab experiments with TurtleBot 2, see Figure 4.1b. The robot has collected the data samples as described in Section 4.3.2, yielding 500 samples in the buffer and 250 samples in the test set.

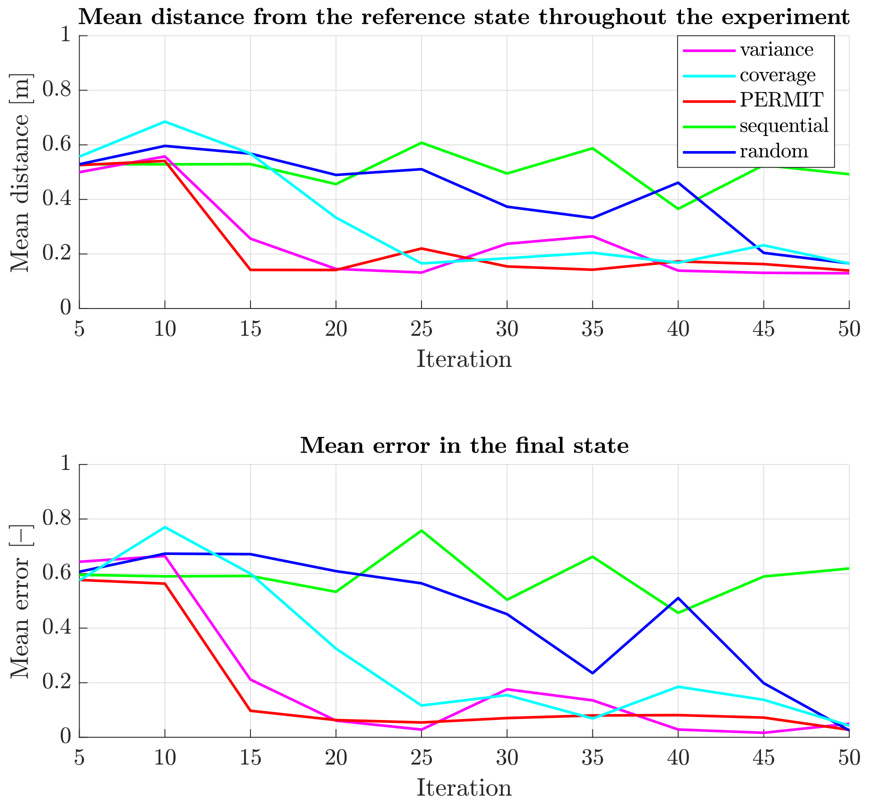


Figure 4.4: Evaluation of the control task executions performed with the real mobile robot.

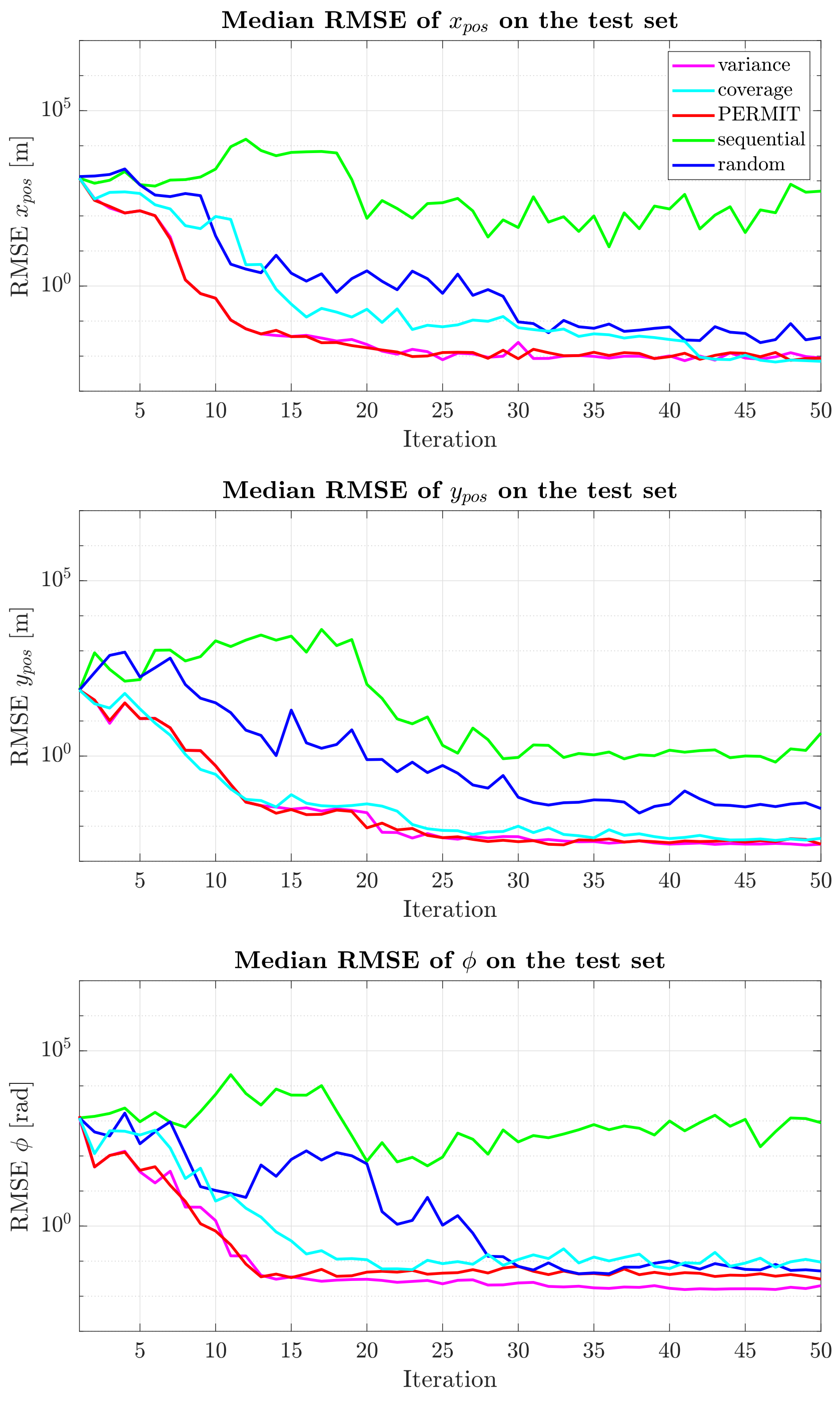


Figure 4.5: Evaluation of the sample-selection methods for modeling all variables in the experiment with the real mobile robot. The number of training samples starts at five in the first iteration and increases by one in each iteration.

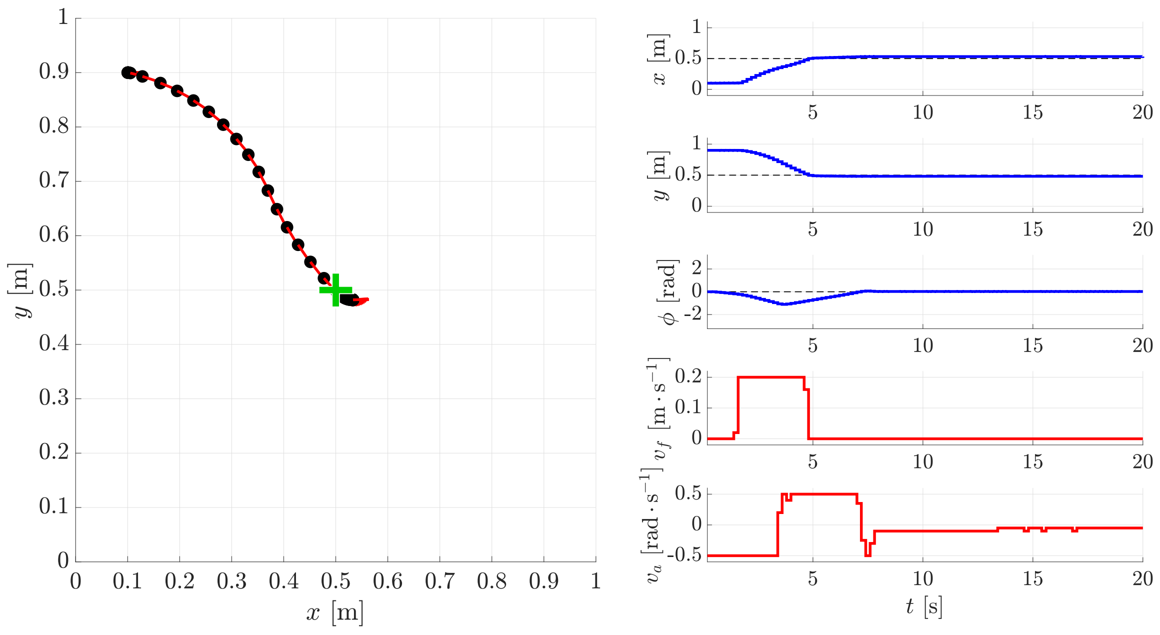


Figure 4.6: Execution of the control task performed on the real robot with an analytic model trained on a data set of only 24 data samples, selected by the PERMIT method. The plot shows the trajectory of the robot (solid black circles) with its orientation (red markers). The reference state is marked by a green cross.

The quality of the models measured by RMSE on the test data set throughout the execution of the model learning algorithm is shown in Figure 4.5 and the quantitative results are summarized in Table 4.1.

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 Figure 4.4. 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.

An example of the control task execution on the real robot is shown in Figure 4.6. 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].

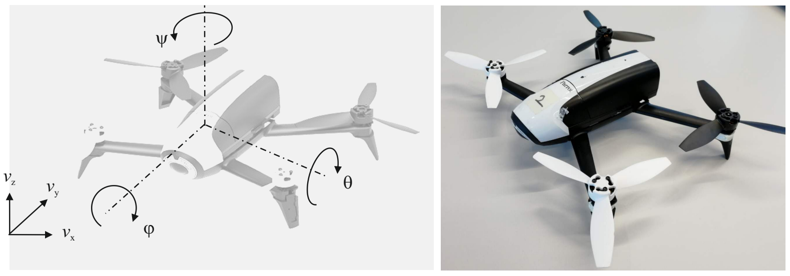
[1]https://github.com/erik-derner/sample-selection/blob/main/TurtleBot_ModelLearning.mp4

4.4. DRONE EXPERIMENTS

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

4.4.1. 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. Figure 4.7 shows a schematic and a photo of the drone.



(a) (b)

Figure 4.7: 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.

4.4.2. 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 in the input–output form $\mathbf{y}_{k+1} = \mathbf{g}(\mathbf{y}_k, \mathbf{y}_{k-1}, \ldots, \mathbf{u}_k, \mathbf{u}_{k-1}, \ldots)$ are constructed for each state variable, 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.

4.4.3. 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 $n_i = 25$.

Table 4.2: 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	$v_x\,[\mathrm{m\cdot s^{-1}}]$	$v_y\,[\mathrm{m\cdot s^{-1}}]$	$v_z\,[\mathrm{m\cdot s^{-1}}]$	θ [rad]	φ [rad]	ψ [rad]
Average RMSE median	variance	2.41×10^{1}	1.06×10^{1}	8.22×10^{1}	4.48×10^{-1}	3.39×10^{-1}	5.91×10^{-1}
	coverage	1.60×10^{1}	3.29×10^{1}	8.07×10^{1}	5.51×10^{-1}	3.90×10^{-1}	6.91×10^{-1}
	PERMIT	7.03×10^{0}	1.16×10^{1}	8.61×10^{1}	3.87×10^{-1}	2.99×10^{-1}	5.86×10^{-1}
	sequential	3.42×10^{6}	2.83×10^{4}	1.11×10^{3}	1.76×10^{0}	1.64×10^{0}	1.81×10^{0}
	random	1.44×10^{1}	2.21×10^{1}	7.80×10^{1}	4.84×10^{-1}	4.18×10^{-1}	6.45×10^{-1}
Average RMSE spread	variance	7.04×10^{6}	3.64×10^{4}	2.93×10^{5}	6.60×10^{-1}	5.74×10^{-1}	4.82×10^{-1}
	coverage	3.25×10^{4}	1.02×10^{7}	1.68×10^{18}	8.67×10^{-1}	8.20×10^{-1}	7.53×10^{-1}
	PERMIT	5.24×10^{3}	3.56×10^{4}	2.93×10^{5}	6.14×10^{-1}	6.27×10^{-1}	5.95×10^{-1}
	sequential	1.28×10^{65}	4.55×10^{22}	4.27×10^{27}	9.43×10^{-1}	1.31×10^{0}	6.03×10^{-1}
	random	2.39×10^{6}	7.56×10^{4}	2.93×10^{5}	7.53×10^{-1}	9.26×10^{-1}	7.66×10^{-1}

4.4.4. Results

The performance of the five sample-selection methods of Section 4.2.2 is shown in Figure 4.8. The quantitative measures are summarized in Table 4.2. 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 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 4.3. 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.

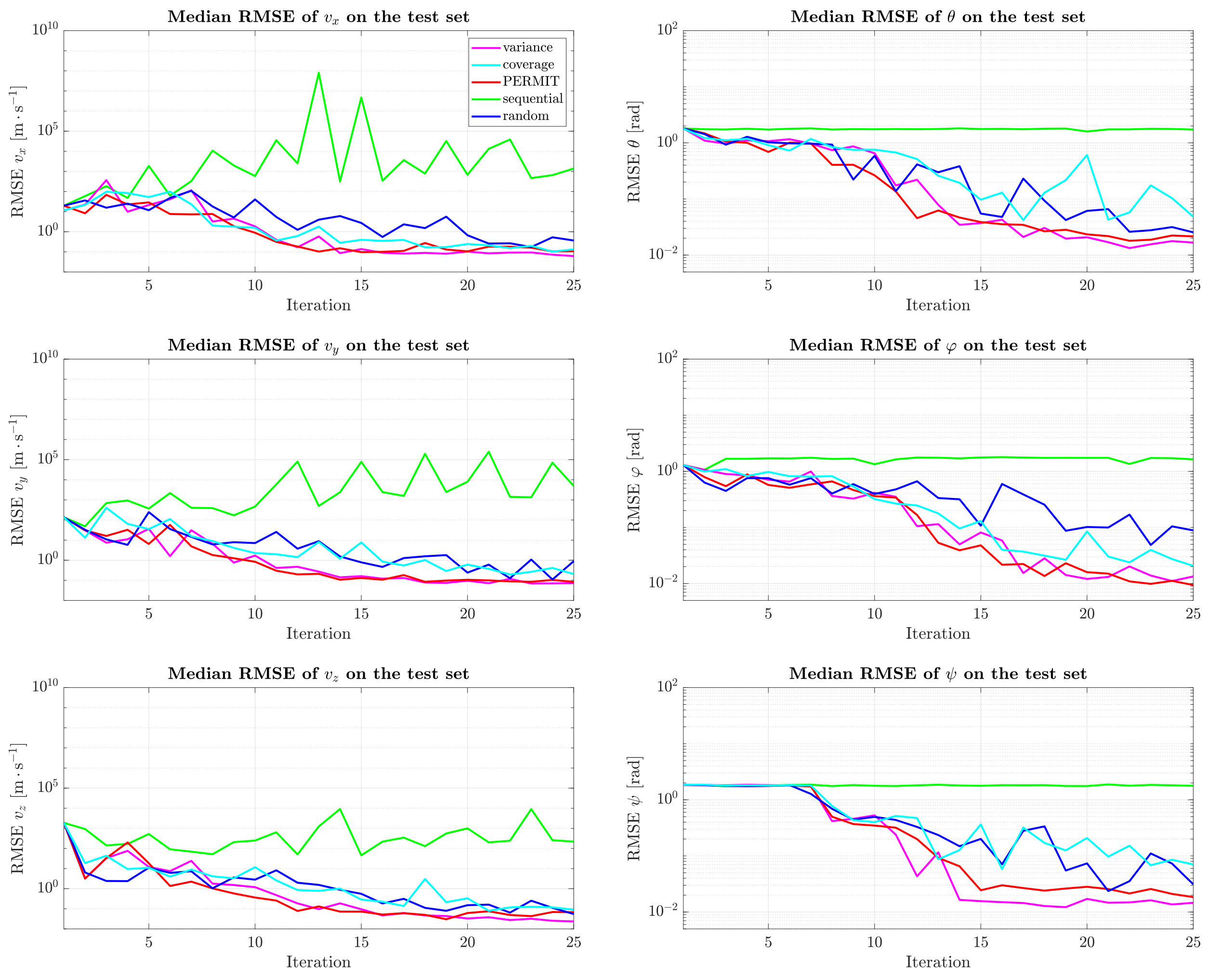


Figure 4.8: Evaluation of the sample-selection methods for modeling all variables in the experiment with the drone. The number of training samples starts at five in the first iteration and increases by one in each iteration.

4.5. Conclusions

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.

5

PHYSICS-AWARE MODEL LEARNING FOR DYNAMIC SYSTEMS

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.

This chapter is an adapted version of the conference paper [46].

5.1. 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 nonlinear model of the robot dynamics, as presented in the surveys [114, 130, 134]. Many of the popular methods such as deep neural networks [107] and locally weighted learning [143] 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 [151], 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 [43, 125, 144].

In this chapter, we extend our previous work [43, 88] 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 chapter 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 [43], and SR using formal constraints [88].
- 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 chapter is organized as follows: Section 5.2 gives an overview of the related work. The model construction method is described in Section 5.3. Section 5.4 presents the results of the experimental evaluation and Section 5.5 concludes the chapter.

5.2. 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 [71]. 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 [76]. Some SR approaches take into account prior knowledge in the form of formal constraints [18, 88]. Another recent SR approach named AI Feynman [140] exploits known properties of the function sought, such as 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 [63], inspired by Hamiltonian mechanics. A neural network is trained to learn and follow the basic laws of physics. Gaussian processes [81, 115] and Koopman operators [1, 25, 103] 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 nonlinear 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 [72, 109]. 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 [88], 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 [43] as well as in SR with formal constraints [88].

5.3. METHOD

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

5.3.1. BASELINE SYMBOLIC REGRESSION

Symbolic regression uses genetic programming to build from data nonlinear models in the following form:

$$y = f(\boldsymbol{\xi}), \tag{5.1}$$

where $\boldsymbol{\xi} \in \mathscr{X} \subset \mathbb{R}^n$ is the model input and $y \in \mathscr{Y} \subset \mathbb{R}$ is its output. The method can be applied to both discrete-time and continuous-time dynamic models. In this chapter, we adapt the model definition described in Chapter 2 to state-space models in continuous time

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \tag{5.2}$$

with the state $\mathbf{x} \in \mathscr{S} \subset \mathbb{R}^s$, the control input $\mathbf{u} \in \mathscr{U} \subset \mathbb{R}^m$, 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 5.4. 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 [71]. 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 [7] and construct the model as a composition of genetically evolved nonlinear functions f_i, called *features*:

$$f(\boldsymbol{\xi}) = \beta_0 + \sum_{i=1}^{n_f} \beta_i f_i(\boldsymbol{\xi}), \tag{5.3}$$

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 $\mathscr{F}$, 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 [87].

5.3.2. 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:

$$f(\boldsymbol{\xi}) = \beta_0 + \underbrace{\sum_{i=1}^{n_p} \beta_i \bar{f}_i(\boldsymbol{\xi})}_{\text{prior features}} + \underbrace{\sum_{i=n_p+1}^{n_f} \beta_i f_i(\boldsymbol{\xi})}_{\text{evolved features}}. \tag{5.4}$$

Features $\bar{f}_i(\boldsymbol{\xi})$ encode the prior knowledge in the form of fixed functions specified by the user in advance, while features $f_i(\boldsymbol{\xi})$ 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 $\bar{f}_i(\boldsymbol{\xi})$ and $f_i(\boldsymbol{\xi})$. 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:

$$c_i^E(\boldsymbol{\xi}) = 0 \quad \forall\, \boldsymbol{\xi} \in \mathscr{C}_i^E \subset \mathscr{X} \tag{5.5}$$

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

$$c_i^I(\boldsymbol{\xi}) \leq 0 \quad \forall\, \boldsymbol{\xi} \in \mathscr{C}_i^I \subset \mathscr{X} \tag{5.6}$$

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 $\mathscr{C}_i^E$ and $\mathscr{C}_i^I$. This yields two sets of samples: D_i^E containing samples for each equality constraint $c_i^E(\boldsymbol{\xi})$ and D_i^I with samples for each inequality constraint $c_i^I(\boldsymbol{\xi})$. The *constraint violation error*, denoted e_c, takes into account both types of constraints. We refer the interested reader to [88] 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.

5.4. 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.

5.4.1. 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 5.4.2.

We use the following data sets for training and evaluation:

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.

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.

Table 5.1: Main SNGP parameters used in the experiments.

Parameter	Symbol	Value
Elementary function set	$\mathscr{F}$	$\{+, -, \times$, 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

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.

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 [43, 44] and is beyond the scope of this work.

5.4.2. 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 5.1. 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.

5.4.3. Mobile Robot

We have recorded the training and test data sets on a real robot TurtleBot 2, see Figure 4.1 in Chapter 4. For convenience, we repeat here that 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:

$$\dot{x}_{pos} = v_f \cos\phi\,, \tag{5.7}$$

$$\dot{y}_{pos} = v_f \sin\phi\,, \tag{5.8}$$

$$\dot{\phi} = v_a\,. \tag{5.9}$$

[1]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:

$$\hat{\dot{x}}_{pos} = f_{\dot{x}_{pos}}(x_{pos}, y_{pos}, \phi, v_f, v_a)\,, \tag{5.10}$$

$$\hat{\dot{y}}_{pos} = f_{\dot{y}_{pos}}(x_{pos}, y_{pos}, \phi, v_f, v_a)\,. \tag{5.11}$$

Modeling of $\dot{\phi}$ is omitted from this work 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 (5.10) and (5.11), all the state and input variables are available to SR, even though they are not needed according to the theoretical models (5.7), (5.8). SR automatically chooses the variables that are useful to capture the properties of the function fitting the data.

PRIOR KNOWLEDGE

The theoretical models (5.7) and (5.8) 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: $x_{pos} \in [-10, 10]\,\mathrm{m}$, $y_{pos} \in [-10, 10]\,\mathrm{m}$, $\phi \in (-\pi, \pi]\,\mathrm{rad}$, $v_f = 0\,\mathrm{m}\cdot\mathrm{s}^{-1}$, $v_a \in [-\pi, \pi]\,\mathrm{rad}\cdot\mathrm{s}^{-1}$ $\rightarrow \dot{x}_{pos} = 0\,\mathrm{m}\cdot\mathrm{s}^{-1}$.

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: $x_{pos} \in [-10, 10]\,\mathrm{m}$, $y_{pos} \in [-10, 10]\,\mathrm{m}$, $\phi = \pi/2\,\mathrm{rad}$, $v_f \in [-0.6, 0.6]\,\mathrm{m}\cdot\mathrm{s}^{-1}$, $v_a = 0\,\mathrm{rad}\cdot\mathrm{s}^{-1} \rightarrow \dot{x}_{pos} = 0\,\mathrm{m}\cdot\mathrm{s}^{-1}$.

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: $x_{pos} \in [-10, 10]\,\mathrm{m}$, $y_{pos} \in [-10, 10]\,\mathrm{m}$, $\phi = -\pi/2\,\mathrm{rad}$, $v_f \in [-0.6, 0.6]\,\mathrm{m}\cdot\mathrm{s}^{-1}$, $v_a = 0\,\mathrm{rad}\cdot\mathrm{s}^{-1} \rightarrow \dot{x}_{pos} = 0\,\mathrm{m}\cdot\mathrm{s}^{-1}$.

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 $\gamma = 5 \times 10^{-2}\,\mathrm{m}\cdot\mathrm{s}^{-1}$ 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.

Table 5.2: 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} $(\mathrm{m \cdot s^{-1}})$
$\hat{x}_{pos}$	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}
$\hat{y}_{pos}$	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}

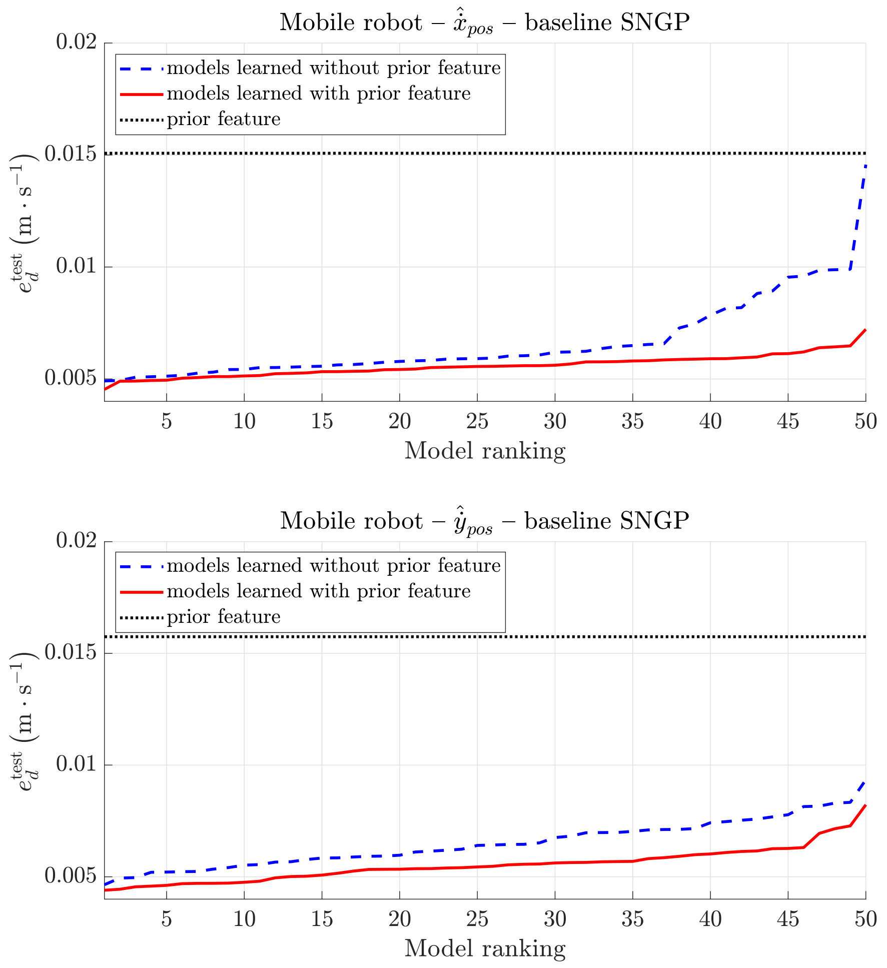


Figure 5.1: Comparison of models from 50 baseline SNGP runs in the mobile robot experiment, sorted by RMSE on the test data set e_d^{test}.

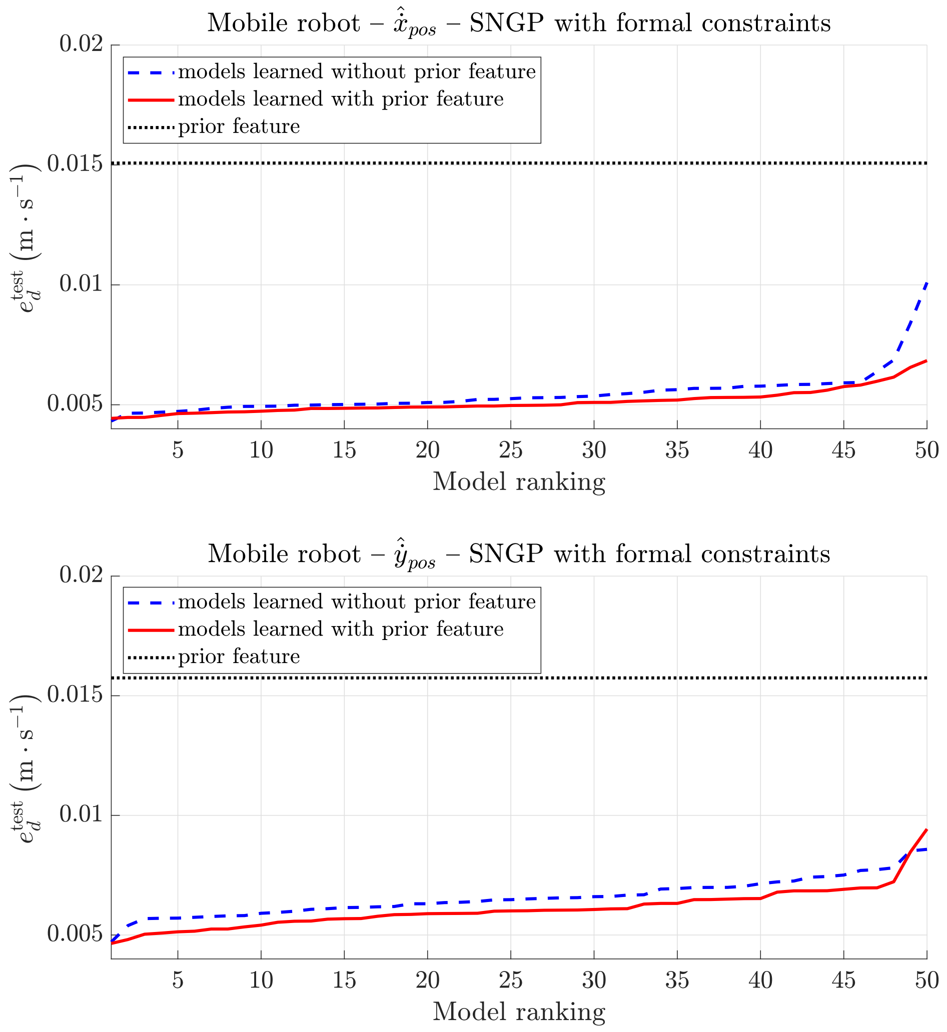


Figure 5.2: Comparison of models from 50 runs of SNGP with constraints in the mobile robot experiment, sorted by RMSE on the test data set e_d^{test}.

Results and Discussion

The results for all cases are summarized in Table 5.2. 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 Figure 5.1 for baseline SNGP and in Figure 5.2 for SNGP with formal constraints. The error of the prior feature itself is shown as a reference.

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 $\hat{x}_{pos}$ and $\hat{y}_{pos}$. We used the Wilcoxon rank-sum test [62] 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 $\hat{x}_{pos}$. Even a more significant improvement is achieved for $\hat{y}_{pos}$ 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 $\hat{x}_{pos}$ and $\hat{y}_{pos}$ 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 $\hat{x}_{pos}$ and $\hat{y}_{pos}$, 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:

$$\begin{aligned}\hat{x}_{pos} = {} & 8.5\times10^{-1} v_f \cos(\phi) - 1.2\times10^{-2} \sin(\sin(x_{pos})) \\ & + 1.3\times10^{-2} \sin(x_{pos})^2 - 7.7\times10^{-3} \cos(\phi)^3 \\ & + 3.7\times10^{-3} (\phi + v_a) + 2.8\times10^{-3} y_{pos} \cos(\phi + v_a) \\ & - 3.2\times10^{-3} (v_f + 2)(\sin(\sin(\phi)) - \cos(\phi)^3 \cos(x_{pos})) \\ & - 1.9\times10^{-3} (\phi + v_a)^2 + 1.8\times10^{-3} \cos(x_{pos})(\phi - 3.1) \\ & + 5.5\times10^{-4} y_{pos} - 4.8\times10^{-4} v_f - 3.1\times10^{-4}\,,\end{aligned} \tag{5.12}$$

$$\begin{aligned}\hat{y}_{pos} = {} & 8.6\times10^{-1} v_f \sin(\phi) - 2.3\times10^{-1} \cos(1.4 v_f) \\ & - 9.0\times10^{-2} v_f \sin(\cos(v_f^2)) - 8.3\times10^{-3} \cos(\phi - 2.8 v_f)^4 \\ & + 5.5\times10^{-3} (\phi + v_a) + 7.6\times10^{-4} x_{pos} - 1.6\times10^{-16} y_{pos}^{18} \\ & - 3.0\times10^{-3} \sin(x_{pos})^2 - 5.2\times10^{-3} \cos(v_a)^9 \\ & - 6.4\times10^{-4} (\phi - 2.8 v_f)^2 (y_{pos} - \sin(x_{pos})) \\ & + 6.1\times10^{-5} (y_{pos} - \sin(x_{pos}))^3 + 2.4\times10^{-1}\,.\end{aligned} \tag{5.13}$$

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 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.

5.4.4. DRONE

We have performed experiments with the Parrot Bebop 2 drone, see Figure 4.7 in Chapter 4. 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 $\dot{v}_y$:

$$\dot{v}_x = g\cos\psi\frac{\tan\theta}{\cos\varphi} + g\sin\psi\tan\varphi - k_D v_x\,, \tag{5.14}$$

$$\dot{v}_y = g\sin\psi\frac{\tan\theta}{\cos\varphi} - g\cos\psi\tan\varphi - k_D v_y\,, \tag{5.15}$$

where the gravitational acceleration $g = 9.81\,\mathrm{m}\cdot\mathrm{s}^{-2}$ and the drag constant $k_D = 0.28\,\mathrm{s}$ 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:

$$\hat{\dot{v}}_x = f_{\dot{v}_x}(v_x, \theta, \varphi, \psi)\,, \tag{5.16}$$

$$\hat{\dot{v}}_y = f_{\dot{v}_y}(v_y, \theta, \varphi, \psi)\,. \tag{5.17}$$

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 $T_s = 0.05\,\mathrm{s}$. Same as for the mobile robot, we used the forward difference to approximate the derivatives.

PRIOR KNOWLEDGE

The theoretical models (5.14) and (5.15) 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 $\dot{v}_x$:

$$\begin{aligned} \bar{f}_1 &= g\cos\psi\tan\theta / \cos\varphi\,, \\ \bar{f}_2 &= g\sin\psi\tan\varphi\,, \\ \bar{f}_3 &= -k_D v_x \end{aligned} \tag{5.18}$$

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

$$\begin{aligned} \bar{g}_1 &= g\sin\psi\tan\theta / \cos\varphi\,, \\ \bar{g}_2 &= -g\cos\psi\tan\varphi\,, \\ \bar{g}_3 &= -k_D v_y\,. \end{aligned} \tag{5.19}$$

This enables the method to tune also the coefficients of the empirical model components through least squares, see Section 5.3.2.

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 $\gamma = 5 \times 10^{-2}\,\mathrm{m \cdot s^{-2}}$ and we choose the model with the lowest e_d^{train}.

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 5.3. 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 $\hat{v}_y$, 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 $\hat{v}_x$, the improvement is substantial: the Wilcoxon rank-sum test returns $p = 2.383 \times 10^{-9}$ for the case with one prior feature and $p = 1.520 \times 10^{-10}$ for the case with three prior features, both compared to the case without any prior feature. Also for $\hat{v}_y$, a significant improvement is achieved with $p = 2.284 \times 10^{-3}$ for one prior feature and $p = 1.040 \times 10^{-5}$ for three prior features, compared to the case with no prior feature.

Table 5.3: 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} $(\mathrm{m \cdot s^{-2}})$
$\hat{v}_x$	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}
$\hat{v}_y$	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}

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.

5.5. 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.

6
Change Detection Using Weighted Features for Image-Based Localization

Autonomous mobile robots are becoming increasingly important in many industrial and domestic environments. Dealing with unforeseen situations is a difficult problem that must be tackled to achieve long-term robot autonomy. In vision-based localization and navigation methods, one of the major issues is the scene dynamics. The autonomous operation of the robot may become unreliable if the changes occurring in dynamic environments are not detected and managed. Moving chairs, opening and closing doors or windows, replacing objects and other changes make many conventional methods fail. To deal with these challenges, we present a novel method for change detection based on weighted local visual features. The core idea of the algorithm is to distinguish the valuable information in stable regions of the scene from the potentially misleading information in the regions that are changing. We evaluate the change detection algorithm in a visual localization framework based on feature matching by performing a series of long-term localization experiments in various real-world environments. The results show that the change detection method yields an improvement in the localization accuracy, compared to the baseline method without change detection. In addition, an experimental evaluation on a public long-term localization data set with more than 10 000 images reveals that the proposed method outperforms two alternative localization methods on images recorded several months after the initial mapping.

This chapter is an adapted version of the journal paper [42].

6.1. INTRODUCTION

Deployment of autonomous mobile robots in industrial and domestic environments is challenging due to the dynamics of these environments. Advanced methods are needed to perform localization and navigation precisely and reliably, despite the changes occurring in the environment.

In this work, we present a novel approach to change detection based on local features and their descriptors. A robot equipped with a camera moves through its environment and detects changes that have occurred with respect to an initial mapping session. Once the robot detects a change, it captures it by decreasing weights assigned to the corresponding features stored in the environment representation. At the same time, stable features that have been detected during subsequent visits to the same place are given higher importance through increasing their weights.

Examples of changes that we consider in our work are moving chairs and items on tables, pictures on computer or TV screens, changing contents of whiteboards and notice boards, opening or closing doors, adjusting blinds in the windows, etc. These changes occur every day in various industrial, domestic, and office environments, as illustrated in Figure 6.1.

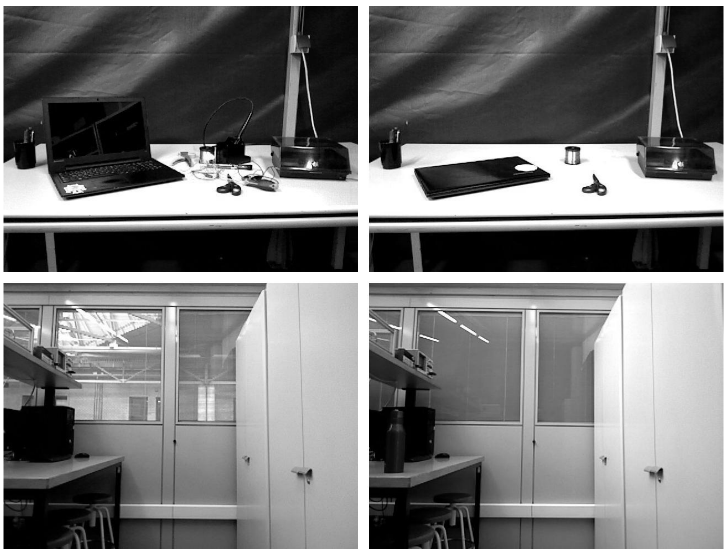


Figure 6.1: Examples of changes that can be detected by the proposed algorithm. Handling these changes appropriately allows for more accurate localization.

The change detection method can be used for robot localization based on place detection, which can be further used to perform more complex tasks such as navigation. It allows the robot to recognize its surroundings more reliably and therefore to perform these tasks more precisely. The advantages of the proposed method are its short learning time, low demand for computational resources, its data efficiency and the low requirements on the sensor hardware.

The chapter is organized as follows. Section 6.2 presents the related research in the field of change detection and localization in dynamic environments. A baseline visual localization framework is introduced in Section 6.3 and the proposed change detection method in Section 6.4, along with its incorporation into the localization framework. Sections 6.5 and 6.6 describe the experimental evaluation. The conclusions and future work are given in Section 6.7.

6.2. RELATED WORK

Dynamically changing environments present a challenge in most robotic navigation contexts. In order to perform stable localization and path-planning, robots must take into account these changes [4]. Change detection and localization in dynamic environments has attracted the interest of many authors and various approaches have been proposed, as surveyed for example in [59] and [101].

Most change detection algorithms are based on object detection and tracking in long-term operation [17, 22, 66, 91]. In [22], a robot patrols an indoor environment and detects movable objects by change detection and temporal reasoning. The objective is to determine how many movable objects are there in the environment and to track their position. The Rao-Blackwellized particle filter and the expectation-maximization algorithm are used to track the objects and to learn the parameters of the environment dynamics. In [91], a service robot is deployed in various indoor environments and a hierarchical map of the environment is maintained that takes into account the changes in the object positions by comparing current object detections to the mapped ones. In [17], the change detection problem is treated through reasoning about observations. Observations are classified considering long-term, short-term, and dynamic features, which correspond to mapped static objects, unmapped static objects, and unmapped dynamic objects, respectively. Short-term features produce local adjustments to the belief about the trajectory of the robot, while long-term features yield global adjustments.

Other works detect changes via correspondences between robot views or images [5, 9, 15, 21, 48, 53, 56, 82, 148]. Full RGB-D views are used in [56] to build a map of the robot world. Changes between successive views are computed to discover the objects (moved areas) and to learn them. Similarly, in [53], a Truncated Signed Distance Function (TSDF) grid and a 3D reconstruction of the environment are maintained. New observations are aligned with the previous ones and included in the new reconstruction. The new reconstruction is compared to the previous one in order to identify dynamic clusters between both reconstructions. Image views are used in [48] to detect changes using Gaussian Mixture Models (GMMs). As GMMs have long computational times, Vertical Surface Normal Histograms provide main plane areas that are discarded in the search for changes. Change detection is accomplished as the difference in the Gaussians generated for two images.

Pointclouds from a LiDAR are compared to an octree-based occupancy map in [148] to obtain a set of changes. Change candidates are computed using the Mahalanobis distance and filtered to eliminate outliers. Authors in [21] proposed a 2D LiDAR-based framework for long-term indoor localization on prior floor plans. The system combines graph-based mapping techniques and Bayes filtering to detect significant changes in the environment. The authors use an iterative closest point-based scan matching to determine the probability that a LiDAR scan related to a trajectory pose corresponds to the currently observable environment. This probability is used to improve the trajectory estimation through the update of the previous nodes.

An approach for mapping and localization dealing with sensor inaccuracies and dynamic environments was presented in [5]. The method based on Extended Kalman Filtering incorporates the measurements of landmark strength in the map. The landmarks between the map and the scene are matched in each step and the landmark strength is updated. Weak landmarks are pruned, while newly observed landmarks are added. A system of visual mapping using an input from a stereo camera was presented in [82]. The method builds and continually updates a metric map of views, represented as a graph. In [9], a method for life-long visual localization using binary sequences from images is proposed. The approach is based on using sequences of images instead of single images for recognizing places. Features are extracted using global LDP descriptors to obtain the binary codes of each image. These binary descriptors are efficiently matched by computing the Hamming distance.

Change detection has also been broadly studied for outdoor environments [2, 111, 117]. Structural change detection from street-view images is performed in [2]. Multisensor fusion SLAM, deep deconvolution networks and fast 3D reconstruction are used to determine the changing regions between pairs of images. In [117], a Bayesian filter is proposed to model feature persistence of road and traffic elements. Single-feature and neighboring-feature information are used to detect changes in feature-based maps and estimate feature persistence. Many works have been devoted to overcoming seasonal changes for outdoor environment navigation [32, 110, 135, 141]. In [110], HOG features and deep convolutional networks are used to compare and match the newly acquired image with a database of images independently of the weather and seasonal conditions. The approach presented in [141] compares different variants of SIFT and SURF feature detectors in the frame of an appearance-based topological localization on panoramic images capturing seasonal changes. Visual words from local features have been used in [73] and [74] to determine the best candidate image given a query image. Visual words are built from the co-occurrence of SIFT features at each location at different times of the day or year. Spatial words (spatial relations between features) are used to verify candidate visual words.

Only a limited number of works have been devoted to long-term localization based on local features in indoor environments [12, 31, 34, 36]. In [31], SURF features are applied to training images in order to learn a new descriptor that is more robust to changes in both indoor and outdoor environments. The work [34] presents an approach based on experiences. An observation is compared with all stored experiences and matched against the most similar one.

The approaches [12] and [36] are inspired by the Atkinson and Shiffrin human memory model [11]. They adopt the concepts of short-term memory (STM) and long-term memory (LTM) to deal with changing environments in long-term localization. In [36], the STM and LTM are represented as finite state machines. Each new feature gets first to the STM and needs to be repetitively detected to be moved to LTM. Features are removed from STM and LTM if they are not detected in successive visits to the same place. In [12], the authors introduce a Feature Stability Histogram (FSH) that registers local feature stability over time through a voting scheme. Both methods employ their own mechanisms to update the respective feature containers. In [36], only LTM is used for matching. In [12], both LTM and STM is used for matching within the FSH and the feature strength is considered. Both approaches use omnidirectional images for experiments, even though the algorithms do not seem to be specifically dependent on that type of images.

Many of the aforementioned algorithms need computationally demanding learning processes and a vast amount of training data [31] or heavy maintenance of 3D map reconstructions [2, 56], while other methods build on spatio-temporal relations [73, 74]. On the contrary, the approach proposed in this chapter relies on local feature detection and matching, which allows it to run in real time on low-cost hardware platforms. It does not require any data-hungry, computationally challenging learning stage to build an initial map of the environment first and then continuously update it during the long-term operation of the robot. No further assumptions need to be made regarding periodic repetitiveness of changes, similar speed of the robot following given trajectories, etc.

Among the two related methods [12] and [36], the proposed method is closest to [36] thanks to a comparable relative simplicity of the approach. However, we do not use the short-term and long-term memory concepts. Instead, we assign weights to the features that capture their stability and therefore also their importance. The feature weights are updated proportionally to the similarity of the feature descriptors, which grants a smoother way of capturing the feature significance than [36].

6.3. VISUAL LOCALIZATION FRAMEWORK

The change detection method proposed in this work can be used for localization or place detection with various algorithms based on local features. In this section, we present a visual localization framework that will serve as a baseline and that will be later extended with the proposed change detection method.

An overview of the method is presented in Figure 6.2. For now, we assume that the change detection module is not active and we introduce the baseline localization framework.

At first, a robot equipped with a camera builds a discrete representation of the environment in the form of a *visual database*, where images are stored together with the robot pose. Note that a robot featuring a self-localization capability in a static environment needs to be employed to build the visual database. The location of the robot can be estimated using e.g. wheel encoders, laser rangefinder readings, inertial measurements, visual odometry, or a combination of these sources.

During the long-term deployment, the robot continuously localizes itself in the environment, based solely on the visual information by using feature matching against the previously built visual database. The matching procedure will be described in detail in Section 6.3.3.

6.3.1. BUILDING THE VISUAL DATABASE

A mobile robot equipped with a camera moves around the environment and records images together with the corresponding robot poses. The visual database consists of records r_i, $i = 1,\dots,N$, which have the following structure:

- a grayscale image I_i, captured by the camera mounted on the robot,
- a set of features F_i detected in the image I_i, where $F_i = (f_i^1, f_i^2, \dots, f_i^{m_i})$,
- a set of descriptors D_i of the features F_i, where $D_i = (d_i^1, d_i^2, ..., d_i^{m_i})$ and the indices $1, 2, \dots, m_i$ match the descriptors with the corresponding features,
- a set of weights W_i of the features F_i, where $W_i = (w_i^1, w_i^2, \dots, w_i^{m_i})$ and the indices $1, 2, \dots, m_i$ match the weights with the corresponding features,
- the coordinates $c_i = (x_i, y_i, \varphi_i)$ representing the pose of the robot.

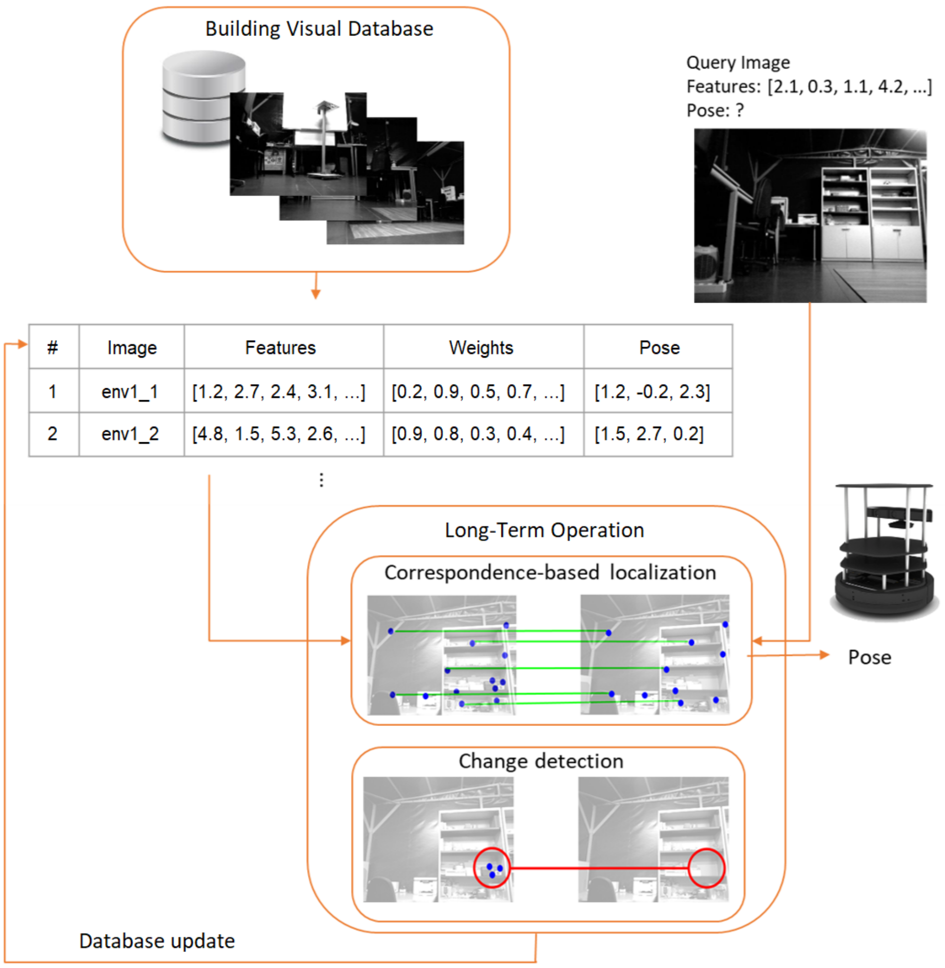

Building Visual Database
Query Image
Features: [2.1, 0.3, 1.1, 4.2, ...]
Pose: ?
# | Image | Features | Weights | Pose
1 | env1_1 | [1.2, 2.7, 2.4, 3.1, ...] | [0.2, 0.9, 0.5, 0.7, ...] | [1.2, -0.2, 2.3]
2 | env1_2 | [4.8, 1.5, 5.3, 2.6, ...] | [0.9, 0.8, 0.3, 0.4, ...] | [1.5, 2.7, 0.2]
Long-Term Operation
Correspondence-based localization
Pose
Change detection
Database update

Figure 6.2: Overview of the visual localization framework. The long-term localization uses a previously built visual database. Query images from the robot are matched against the visual database and the closest match determines the pose of the robot. The change detection module monitors the changes in the matched images and updates the visual database when it detects a difference.

A robust feature detector and descriptor need to be employed to detect the features and calculate their compact representation. The feature detector and descriptor can be selected by the user. As suggested in [12] and [36], we have chosen Speeded-Up Robust Features (SURF) [14]. To verify this choice, an experimental evaluation of different features is given in Section 6.6.5.

6.3.2. WEIGHTED FEATURES

The weights W_i of features F_i are introduced to capture the importance of individual features in each database record r_i. All weights are in the range $w_i^j \in [0,1]$. The weights are initialized at 0.5 for all features that have not been assigned any weight so far.

In the baseline localization method, the weights retain their initial values throughout the algorithm runtime. When updating the visual database based on change detection, the weights capture the stability and therefore also the reliability of the feature – the features with the highest weights are the most stable and reliable ones.

The weights capture the information whether the image patch represented by the feature (and its descriptor) is still present in the scene. This is also why the initial weight of a feature is set to the half of the interval of possible values – at first, it is not known whether the feature is stable (important) or not. The weight update process through change detection will be detailed in Section 6.4.

6.3.3. CORRESPONDENCE-BASED LOCALIZATION

Using the previously constructed visual database, the robot can localize itself using a real-time feed of images from its camera that we refer to as the *query images*. The following steps are performed to localize the robot:

1. Capture a grayscale image I_q – the query image.

2. Run a feature detector and descriptor on the query image. The set of descriptors on the query image I_q is denoted as $D_\mathrm{q} = \{d_\mathrm{q}^1, d_\mathrm{q}^2, \ldots, d_\mathrm{q}^{m_\mathrm{q}}\}$.

3. Match the set of descriptors D_q found in the query image against the sets of descriptors D_i corresponding to each database record r_i using a standard matching algorithm [100].

4. Report the pose of the robot as the pose c_{i^*} stored with the database record r_{i^*}, which achieved the highest weighted correspondences ratio among all records r_i in the database.

The index i^* of the database record r_{i^*} with the highest weighted correspondences ratio is determined by the following equation:

$$i^* = \operatorname*{argmax}_i \left(\frac{\sum_{j=1}^{m_i} p_{\mathrm{q},i}^j w_i^j}{\sum_{j=1}^{m_i} w_i^j} \right), \tag{6.1}$$

where the binary variable $p_{\mathrm{q},i}^j$ captures whether the feature f_i^j from F_i appears in the tentative matches between the query image I_q and the database image I_i. Tentative matches refer to a preliminary matching of descriptors [100] that can be verified through a model estimation method such as RANSAC [57]. This means that we are searching for a database record that has the largest proportion of feature descriptors matched with the feature descriptors found in the query image, giving more importance to the features with higher weights.

We chose to keep the baseline localization method as simple as possible to clearly demonstrate the impact of using weighted features within change detection. However, the proposed method can be plugged into more sophisticated methods as well. For instance, even though we use linear search for simplicity, more advanced techniques can be employed, such as k-d trees to store feature descriptors, allowing for a substantial speed-up for large databases [36]. Other approaches, e.g. visual vocabulary [131] or hierarchical k-means [116], can be used to further improve the performance.

6.4. CHANGE DETECTION METHOD

We propose a method for change detection that improves the long-term autonomy of mobile robots through maintaining an accurate, up-to-date representation of the environment. The essence of the method is in learning the scene regions that are stable, distinguishing them from regions that change. This task is performed through feature-based change detection and results in a representation robust to changes in the environment. In the following text, we present the change detection method and show how it extends the baseline localization framework described in Section 6.3.

6.4.1. DETECTING CHANGES

The change detection algorithm is based on the comparison of feature descriptors. We define a similarity measure between two descriptors d and d' based on their Euclidean distance:

$$s(d, d') = \frac{1}{1 + ||d - d'||_2}. \tag{6.2}$$

This definition follows a standard approach to convert a distance measure to a similarity measure [16]. The similarity measure takes values between 0 and 1 with higher values indicating more similar descriptors. Using Euclidean distance for comparing SURF descriptors is suggested in [14] as one of the standard options.

The outline of the change detection algorithm is as follows:

1. Based on the pairs of tentative correspondences found by the matching algorithm, use MSAC [138] to estimate the transformation between the query image I_q and the best-match database image I_{i^*}.
2. Transform the positions of the features F_{i^*} in the best-match database image I_{i^*} to the coordinate frame of the query image I_q, yielding a set of transformed features $\bar{F}_{i^*}$.
3. Calculate the descriptors $\bar{D}_{i^*} = \{\bar{d}^1_{i^*}, \bar{d}^2_{i^*}, \dots, \bar{d}^{m_{i^*}}_{i^*}\}$ of the transformed features $\bar{F}_{i^*}$ in the query image I_q.
4. Calculate the similarity s_j between the descriptors $d^j_{i^*}$ corresponding to the features $f^j_{i^*}$ in the database image I_{i^*} and the descriptors $\bar{d}^j_{i^*}$ of their projections $\bar{f}^j_{i^*}$ in the query image I_q.

To calculate s_j, the similarity measure (6.2) is adapted to the following form:

$$s_j = \frac{1}{1 + \left\| d^j_{i^*} - \bar{d}^j_{i^*} \right\|_2} \text{ for } j = 1, \dots, m_{i^*}. \tag{6.3}$$

The change detection is therefore based on computing the similarity measure between the descriptors $d^j_{i^*}$ calculated on the best-match database image I_{i^*} and their transformed counterparts $\bar{d}^j_{i^*}$ calculated on the query image I_q. Note that this is different from using the features F_q and their descriptors D_q detected in the query image for comparison.

6.4.2. WEIGHTS UPDATE

The weights of the features in the best-match database image are updated using the similarity values s_j. The weights are updated proportionally to the similarity between the descriptors $d_{i^*}^j$ calculated on the best-match database image I_{i^*} and their transformations $\bar{d}_{i^*}^j$ calculated on the query image I_q:

$$\forall j \in \{1, \ldots, m_{i^*}\} : w_{i^*}^j \leftarrow \min(C s_j w_{i^*}^j, 1) \tag{6.4}$$

As the similarity measure s_j takes values between 0 and 1, we choose $C = 2$ so that the weight remains constant if the similarity s_j is equal to 0.5.

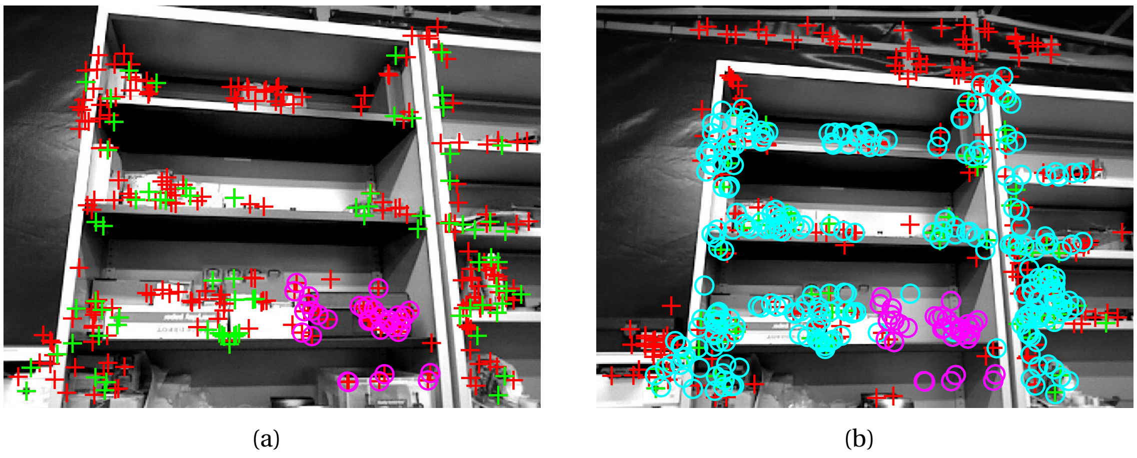


Figure 6.3: A database image (a) and a query image with one object missing on the third shelf from the top (b). The crosses represent the features in both images. Tentative correspondences found by the matching algorithm are shown in green. The cyan circles show the transformations of the features from the database image to the query image. The magenta circles show the features that were identified as a change.

Figure 6.3 shows a scene on which we illustrate the principle of the change detection algorithm. An item, e.g. a toolbox, has been removed from one of the shelves after building the visual database. The change detection algorithm transforms the features from the database image to the query image and calculates their SURF descriptors. They are then compared to the corresponding SURF descriptors in the database image. Since the descriptors in the region of the toolbox have low similarity, their weights are decreased. Note that as a by-product, some unstable features may have their weights decreased as well.

6.4.3. LONG-TERM OPERATION

The localization framework presented in Section 6.3 can now be extended by change detection with feature weight updates. An overview of the long-term localization with the change detection module is shown in Figure 6.2. In the long-term operation, the robot continuously localizes itself in the environment and maintains its visual database up to date by incorporating changes detected in the environment.

An important requirement that makes the change detection method efficient is that wrong matches (localization failures) need to be avoided as much as possible, because incorporating false-positive changes in wrongly matched database records decreases the quality of the visual database. To avoid this, we introduce three conditions that serve as a *confidence criterion*: a spatial condition, a temporal condition, and a sufficient number of correspondences. Only if all three conditions are met, the change detection module is run and the visual database is updated. In this way, the chance of incorporating false change detections on wrongly matched images is minimized.

SPATIAL CONDITION

The spatial condition relates the 'physically' closest database records (pose-wise) and the closest matches found by the localization algorithm (descriptor-wise), with reference to the best-match record. It exploits the assumption that images taken from nearby positions have some common features. Simply put, it is met only if the poses of the most similar matches are not too far from each other, taking into account the density of the database records. The spatial condition is evaluated through the following procedure:

1. Determine n_s most similar matches in the visual database for the query image, where n_s is a user-specified parameter. Sort them in descending order of the weighted correspondences ratio, which is calculated as the argument of (6.1).

2. Calculate the Euclidean distances e_i, $i = 2, \ldots, n_s$, between the best match and the successive $(n_s - 1)$ most similar matches. For this purpose, the distances are calculated in the (x, y)-space of the robot (omitting the angle φ).

3. Calculate the mean Euclidean distance m_s:

$$m_s = \frac{1}{n_s - 1} \sum_{i=2}^{n_s} e_i \, . \tag{6.5}$$

4. Similarly as in Step 2, calculate the Euclidean distances e'_j between the best match and $(n_r - 1)$ database records closest to the best match in the (x, y)-space of the robot, where n_r is a user-specified parameter.

5. Calculate the reference mean Euclidean distance m_r:

$$m_r = \frac{1}{n_r - 1} \sum_{j=2}^{n_r} e'_j \, . \tag{6.6}$$

6. Compare m_s with m_r. If $m_s < m_r$, the poses of the most similar matches are close enough to each other and the spatial condition is met.

Typically, the number of closest reference records n_r is set to be several times larger (e.g. 5×) than n_s. Note that the reference means m_r can be pre-calculated offline for all database records for efficiency.

Temporal Condition

The temporal condition verifies whether the distance between the current location and the previous location is smaller than a given threshold δ. It takes into account the physical limitations of the robot and discards unreliable matches in cases when the robot appears to have moved further than it possibly could. The temporal condition is tested only if the spatial condition was met both for the current and for the previous image, which allows for recovery after a localization failure.

Sufficient Number of Correspondences

As described in Step 1 of the change detection algorithm in Section 6.4.1, we use MSAC [33] to estimate the transformation between the query image and the best-match database image. In this way, we also obtain the information on which features are considered inliers and which are outliers. A small number of inlier correspondences indicates that the match is not very reliable. Therefore, if the number of corresponding feature pairs classified as inliers is smaller than a given threshold θ, the weights are not updated.

6.4.4. Limitations

The proposed method adjusts the importance of features in the database images through their weights. This approach relates all subsequent visits of a place to the first mapping session. If the appearance of a place changes (almost) completely, there are none or only a very few features that can be matched between the current view and the appearance of the place captured during the visual database creation. In such cases, the query image cannot be matched correctly. For instance, if applied to outdoor environments, the method is in general not able to deal with seasonal changes. Nevertheless, unlike other authors [12, 36], we have decided not to include feature addition to the algorithm. The motivation behind this design choice is that new features might come, for instance, from a temporary occlusion or a severe change of lighting conditions. Distinguishing such situations from permanent changes usually means introducing more parameters to the method. Our aim is to keep the long-lasting features from the initial scanning, as we presume that the main structures in most of the scenes remain stable over long periods of time.

As we only use single camera images and no information about the depth is needed, we are limited to the use of planar perspective feature projection (Section 6.4.1). The projection works well if the viewpoints (robot poses) in the visual database are similar to those in the query images, which is the case in the data sets used in the experiments. If the robot gets too far from the records present in the visual database, the viewpoint changes significantly and the reprojection of features does not work well. As a consequence, even though it would be possible to update also the adjacent images in the visual database with the detected changes, we do not perform this update as the benefit would be suppressed by the errors incurred by the reprojection.

While the algorithm allows for adding new images to the database at any point, new database records need to come with the information about the pose of the robot, see Section 6.3. Such information may not be available to the robot performing the long-term operation, as the only sensor it needs to have in our setting is a camera.

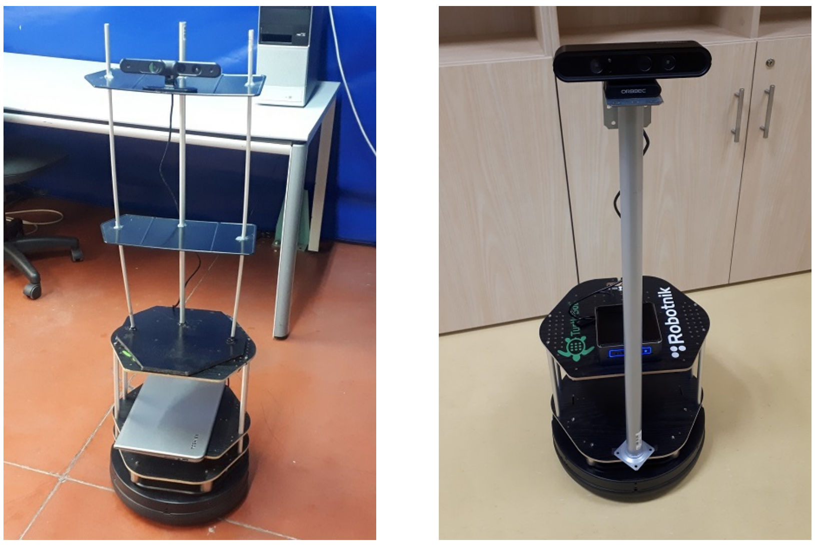
6.5. Evaluation on Our Data Sets

For the experimental evaluation, we have chosen TurtleBot 2 as a widely used and affordable platform, equipped with an RGB-D camera and an on-board computer with a performance comparable to standard laptops. Note that even though the camera is capable of recording depth images, the depth information is not used in our method, as we only use grayscale images. We have used two TurtleBot robots to evaluate the robustness of the method using two different cameras in various environments:

1. TurtleBot 2 equipped with a camera Asus Xtion PRO LIVE, used in experiments at the Robotics Lab, Carlos III University in Madrid, Spain;

2. TurtleBot 2 equipped with a camera Orbbec Astra Pro, used in experiments at the Czech Institute of Informatics, Robotics, and Cybernetics, Czech Technical University in Prague, Czech Republic.

In both cases, we attached an additional structure to the robot to fix the camera at a higher position, see Figure 6.4. The pose of the robot was captured through odometry based on wheel encoders. We used floor markers and manual correction of the measured poses to ensure sufficient ground truth precision.



(a) (b)

Figure 6.4: Mobile robots TurtleBot 2 used in the experiments: (a) at the Carlos III University in Madrid, (b) at the Czech Technical University in Prague.

6.5.1. Data Sets

While several data sets featuring city streets, seasonal changes and other outdoor environments are available [141], the number of available long-term indoor data sets is limited. With the exception of the STRANDS Witham Wharf data set [85], which we evaluate in Section 6.6, there are, to the best of our knowledge, no public data sets capturing indoor dynamic environments which would be well suited for the type of changes that we focus on in this work. To allow for a detailed analysis and better insight into the method's performance, we have recorded our own data sets from four indoor environments at the Leganés campus of the Carlos III University in Madrid and at the Czech Institute of Informatics, Robotics, and Cybernetics in Prague.

The data sets in each of the environments – *Lab*, *Classroom*, *Hall*, and *Office* – consist of multiple sequences. The sequences were recorded on different days and at different times of the day, capturing various changes in the environment (moving chairs and items on the desks, changing the picture on computer screens, opening and closing window blinds, etc.). We have also recorded new sequences after eight months in the *Lab*, *Classroom*, and *Hall* environments to evaluate the long-term performance of the method on a longer time span.

6.5.2. Experimental Setup

At first, we have constructed a visual database for each environment. Our visual databases are sparser than the query sequences to make the database images distinctive and avoid multiple records from the same place. Therefore, the visual databases typically contain fewer images than the query sequences. The playback of the recorded query sequences served as a stream of query images in real time. Each query image was matched with the most similar image in the visual database and the pose associated with the most similar database image was returned as the pose of the robot. If the confidence conditions (Section 6.4.3) were met, the feature weights of the best match record in the visual database were updated based on the detected changes.

In all experiments, we have used the following default configuration of the confidence criterion: the number of closest samples was set to $n_s = 2$ and the number of reference samples $n_r = 10$. The temporal difference tolerance was set to $\delta = 0.5\,\mathrm{m}$ and the minimum number of correspondences was $\theta = 10$. We have empirically evaluated that the default values worked well in all performed experiments. However, they may be adjusted for improved performance on data sets with substantially different properties. Optimization and further analysis of the parameters will be performed as a part of our future work.

We have compared the root-mean-squared (RMS) localization errors on the query sequences matched against the original visual databases and against the visual databases that have been updated by executing localization with change detection on the query sequences. For all images in all query sequences, a maximum of 20 % of the image area has changed with respect to the visual database and with respect to the other query sequences.

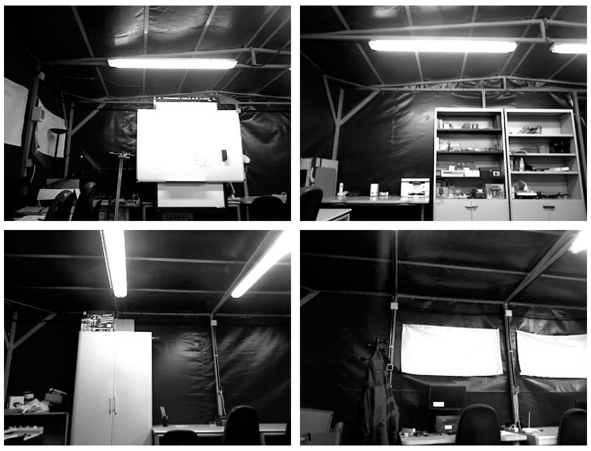
The localization errors are calculated for each pair of a query image and the matched database image as an ℓ^2 norm (Euclidean distance) between the ground truth robot pose of the query image and the robot pose stored with the matched database image. The robot orientation angle is also included in the robot pose vector and it is wrapped to yield a maximal difference of π rad. Note that the units of the localization RMS error are not meters, as the pose vector combines values in meters and radians.

6.5.3. Results

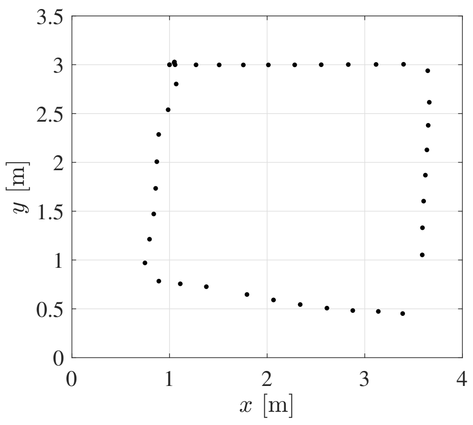
In this section, we present the results of the method on four indoor environments: *Lab*, *Classroom*, *Hall*, and *Office*. For each of them, we have randomly chosen different combinations of the query sequences for updating the visual database to demonstrate the method performance in various scenarios.

Lab Environment

The *Lab* data set was recorded at the Carlos III University in Madrid. The changes in the environment that appear in this data set include moving up to 3 chairs, altering the content on the whole area of the whiteboard, moving up to 10 objects on desks, and moving up to 5 items on the shelves of the cabinets.



(a)



(b)

Figure 6.5: *Lab* environment: (a) examples of images, (b) the trajectory travelled.

First, we have created the visual database L-DB for a trajectory of an approximately square shape, see Figure 6.5. We have recorded three other sequences, L-Q1, L-Q2, and L-Q3, which serve as the query sequences. The sequence L-Q4 was recorded eight months later. Table 6.1 summarizes the properties of the sequences and Table 6.2 presents the results.

Table 6.1: Properties of the image sequences used in the Lab environment.

Image sequence	Number of images	Distance travelled
L-DB (database)	41	10.0 m
L-Q1 (query)	89	10.5 m
L-Q2 (query)	85	10.2 m
L-Q3 (query)	84	10.0 m
L-Q4 (query)	97	10.4 m

The localization accuracy on the query sequence L-Q3 improves when the visual database L-DB is updated by changes detected on the sequence L-Q1. If the visual database is afterward updated also on the sequence L-Q2, the localization accuracy on the query sequence L-Q3 is further improved.

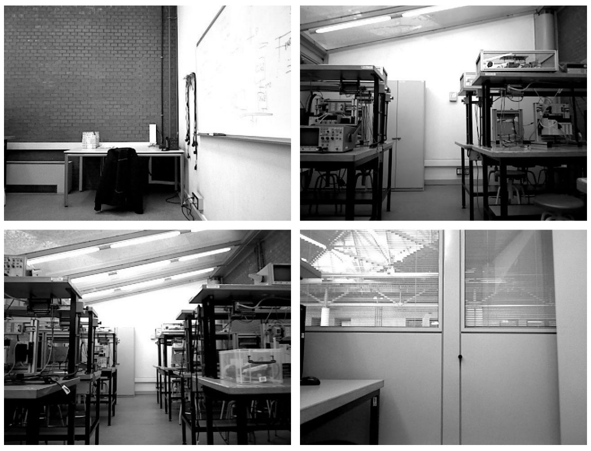
Table 6.2: Localization RMS errors on different query sequences in the Lab environment. Sequences evaluated on a database updated with changes are shown in bold.

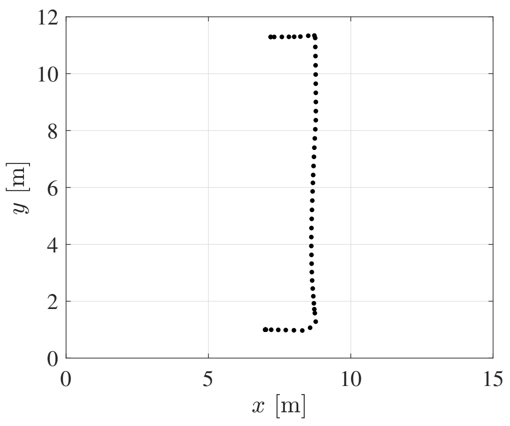
Visual database	Updated on	Query sequence	Localization RMS error
L-DB	–	L-Q3	0.56
L-DB	**L-Q1**	**L-Q3**	**0.46**
L-DB	**L-Q1, L-Q2**	**L-Q3**	**0.41**
L-DB	–	L-Q4	1.25
L-DB	**L-Q1**	**L-Q4**	**1.20**

The number of changes that have occurred in the environment during the long time span before recording the new query sequence L-Q4 was significant. Despite that, the localization RMS error has improved on the query sequence L-Q4 when it was evaluated on the visual database L-DB updated by the sequence L-Q1, compared to evaluating it on the original visual database L-DB. The sequence L-Q1 captures changes such as moved chairs and objects on tables, which helps to build a more robust representation of the environment. The stable elements in the environment are assigned higher weights, which allows for a more accurate localization even after a long period of time.

Classroom Environment

The *Classroom* data set, featuring e.g. desks with items being moved on them, a whiteboard with changing content, and adjustable window blinds, was also recorded at the Carlos III University in Madrid. We have first created the visual database C-DB, see Figure 6.6. Two query sequences, C-Q1 and C-Q2, were recorded several days after the visual database C-DB, and another query sequence, C-Q3, was recorded eight months later. Table 6.3 summarizes the properties of the sequences and Table 6.4 presents the results.


(a)



(b)

Figure 6.6: *Classroom* environment: (a) examples of images, (b) the trajectory travelled.

The localization task was more challenging in the *Classroom* environment. In the case of the query sequence C-Q2 evaluated on the visual database C-DB updated on C-Q1, the localization accuracy remains the same as for the evaluation on the original C-DB. However, for

Table 6.3: Properties of the image sequences used in the Classroom environment.

Image sequence	Number of images	Distance travelled
C-DB (database)	62	13.5 m
C-Q1 (query)	100	13.6 m
C-Q2 (query)	101	13.3 m
C-Q3 (query)	117	13.3 m

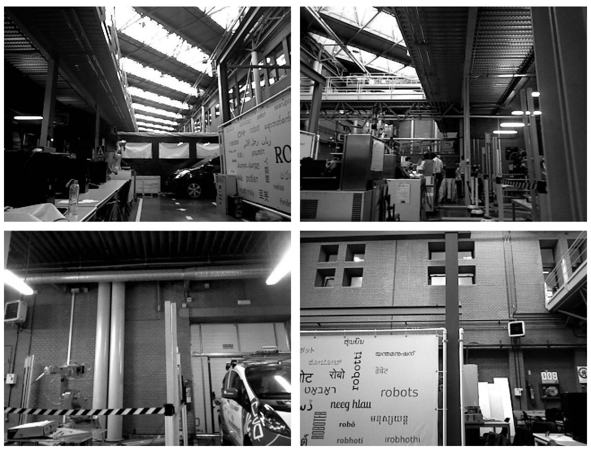
Table 6.4: Localization RMS errors on different query sequences in the Classroom environment. Sequences evaluated on a database updated with changes are shown in bold.

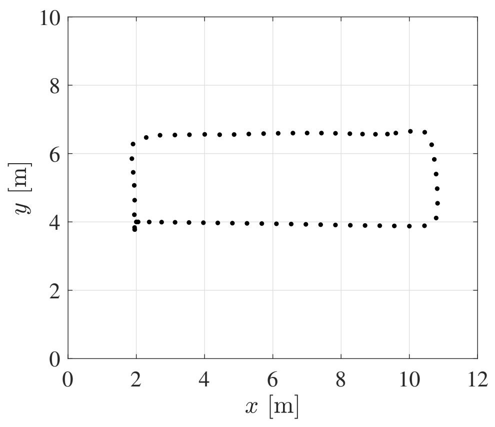
Visual database	Updated on	Query sequence	Localization RMS error
C-DB	–	C-Q2	0.39
C-DB	**C-Q1**	**C-Q2**	**0.39**
C-DB	–	C-Q3	1.02
C-DB	**C-Q1**	**C-Q3**	**0.96**

the new query sequence C-Q3, updating the visual database helps to decrease the localization error.

Hall Environment

The *Hall* data set, resembling an industrial environment, was also recorded at the Carlos III University in Madrid. We have first created the visual database H-DB for a rectangular trajectory, see Figure 6.7. We have recorded two other sequences, H-Q1 and H-Q2, which serve as the query sequences. Eight months later, we have added another sequence H-Q3. Table 6.5 summarizes the properties of the sequences and Table 6.6 presents the results.


(a)



(b)

Figure 6.7: *Hall* environment: (a) examples of images, (b) the trajectory travelled.

Taking into account the spacing between consecutive database images (see Figure 6.7b), the localization results using the initial visual database are already accurate, leaving little space

Table 6.5: Properties of the image sequences used in the Hall environment.

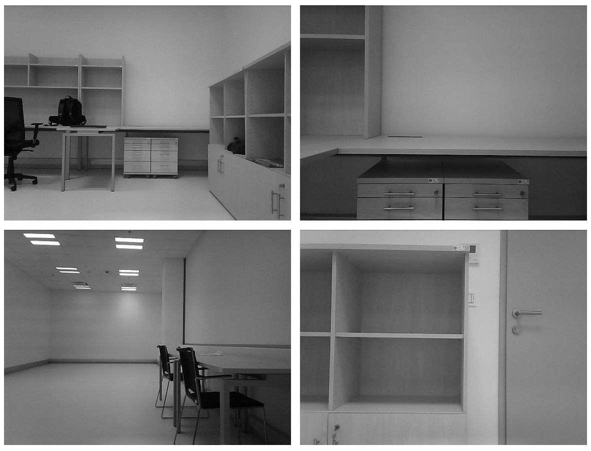
Image sequence	Number of images	Distance travelled
H-DB (database)	58	22.5 m
H-Q1 (query)	67	7.1 m
H-Q2 (query)	59	6.9 m
H-Q3 (query)	53	6.2 m

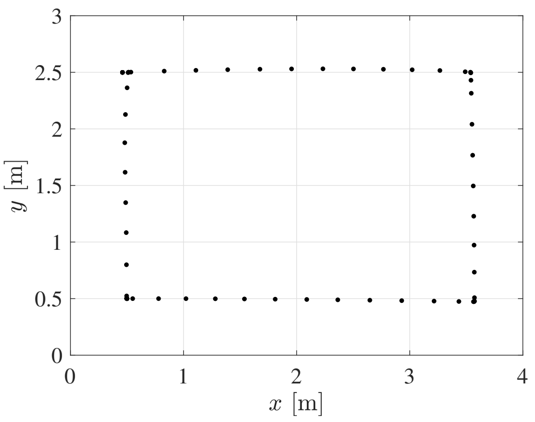
Table 6.6: Localization RMS errors on different query sequences in the Hall environment. Sequences evaluated on a database updated with changes are shown in bold.

Visual database	Updated on	Query sequence	Localization RMS error
H-DB	–	H-Q1	0.20
H-DB	**H-Q2**	**H-Q1**	**0.19**
H-DB	–	H-Q2	0.16
H-DB	**H-Q1**	**H-Q2**	**0.15**
H-DB	–	H-Q3	1.80
H-DB	**H-Q1**	**H-Q3**	**1.64**
H-DB	**H-Q2**	**H-Q3**	**1.65**

for improvement. Nevertheless, an improvement was achieved in both cases – for the query sequence H-Q2 evaluated on the visual database H-DB updated with H-Q1 and for the query sequence H-Q1 evaluated on the visual dabatase H-DB updated with H-Q2. Within the 8 months, the *Hall* environment has undergone substantial changes. The visual database H-DB updated with any of the two sequences H-Q1 or H-Q2 allows for a more accurate localization on the new query sequence H-Q3.

OFFICE ENVIRONMENT


(a)



(b)

Figure 6.8: *Office* environment: (a) examples of images, (b) the trajectory travelled.

The *Office* data set was recorded at the Czech Technical University in Prague. The office undergoes changes such as opening and closing cabinet doors, changing positions of the chairs, or replacing items on the desks. We have first created the visual database O-DB for a rectangular trajectory, see Figure 6.8. We have recorded four other sequences, O-Q1, O-Q2, O-Q3, and O-Q4, which serve as the query sequences. Table 6.7 summarizes the properties of the sequences and Table 6.8 presents the results.

Table 6.7: Properties of the image sequences used in the Office environment.

Image sequence	Number of images	Distance travelled
O-DB (database)	67	10.2 m
O-Q1 (query)	119	9.7 m
O-Q2 (query)	116	10.2 m
O-Q3 (query)	116	10.3 m
O-Q4 (query)	111	10.1 m

Table 6.8: Localization RMS errors on different query sequences in the Office environment. Sequences evaluated on a database updated with changes are shown in bold.

Visual database	Updated on	Query sequence	Localization RMS error
O-DB	–	O-Q1	0.74
O-DB	**O-Q2**	**O-Q1**	**0.72**
O-DB	–	O-Q2	0.93
O-DB	**O-Q3**	**O-Q2**	**0.90**
O-DB	–	O-Q3	0.94
O-DB	**O-Q1**	**O-Q3**	**0.90**
O-DB	–	O-Q4	0.97
O-DB	**O-Q2**	**O-Q4**	**0.91**

The *Office* data set proves to be more difficult in terms of precise localization. The environment contains large uniform and textureless areas and also repetitive instances of identical objects, which makes the feature-based localization more challenging. Nevertheless, the change detection method results in improved localization accuracy.

6.5.4. Discussion

The change detection method reduces the localization (pose estimation) RMS error by 27 % in the *Lab* environment and by 9 % in the *Hall* environment. These two environments are rich in distinctive features, which allows for a better performance of the feature-based localization method. Therefore, localization can also better benefit from the change detection method. The other two environments, *Classroom* and *Office*, have shown to be more difficult for localization. However, the change detection method could still improve the localization accuracy by 6 % in both cases. The change detection method has proven to work even when dealing with a long time span in the case of all three environments *Lab*, *Classroom*, and *Hall*, where we have recorded new sequences several months after the first experiments.

The method runs in real time. In our experiments, we were capturing an image every second. The processing time of a single query image is approximately 250 ms on a standard computer (Intel Core i7-4610M @ 3.0 GHz, 16 GB RAM) for visual databases of less than 100 images. Methods allowing speed-up on larger databases can be employed, as discussed in Section 6.3.3. Such extensions are beyond the scope of this work.

6.6. Evaluation on Public Data Set

In addition to the experiments with our own data sets, we evaluate the method on a public data set Witham Wharf from the STRANDS project [85]. The data set is intended for RGB-D localization in changing environments and therefore, it suits well a long-term indoor localization experiment with our method. Furthermore, we use the data set to compare the performance of our method to two alternative methods for feature-based localization [35, 36]. Finally, we analyze the impact of using different feature detectors and descriptors within our method.

6.6.1. Data Set Overview

The data set contains images captured by a mobile robot every 10 minutes at eight locations in an open-plan office. It is divided into a training set recorded over the period of one week and three test sets, which were recorded on three different days throughout the following months. The training set consists of 1008 images for each of the eight locations, while each test set contains 144 images for each location. We omit the available depth information and convert the RGB images to grayscale images, as in Section 6.5.

To form the visual database, we take the first image for each location from the training set, see Figure 6.9. All the remaining images from the training set are used to build the query sequence S-Q1. Each test set corresponds to one of the query sequences S-Q2, S-Q3, and S-Q4, see Table 6.9.

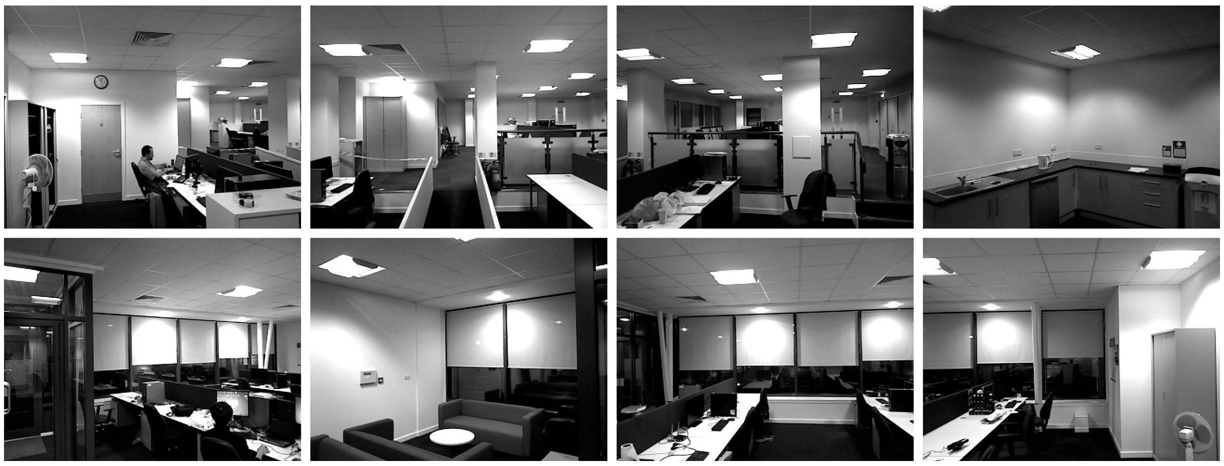


Figure 6.9: Initial images (S-DB) taken from each of the eight locations in the STRANDS Witham Wharf data set.

Table 6.9: Summary of the STRANDS Witham Wharf data set properties.

Image sequence	Number of images	Date of recording
S-DB (database)	8	10 November 2013
S-Q1 (query)	8056	10–16 November 2013
S-Q2 (query)	1152	17 November 2013
S-Q3 (query)	1152	2 February 2014
S-Q4 (query)	1152	14 December 2014

6.6.2. Method Performance

This experiment compares the baseline localization algorithm, where the change detection module is disabled, with localization using change detection. As the data set consists of eight discrete locations rather than images from trajectories (like in Section 6.5), we use a different evaluation metric than for our data sets. We calculate the localization accuracy as the fraction of images that were assigned to the correct ground truth location among all evaluated images. The nature of the data set also does not allow for the use of the spatial and temporal conditiona in the confidence criterion. The confidence criterion therefore relies on the minimum number of correspondences, which was set to $\theta = 10$.

The localization algorithm with change detection takes S-DB as the visual database and runs first on the long query sequence S-Q1 (training set). Then, retaining the feature weights after processing S-Q1, it is executed on the query sequences S-Q2, S-Q3, and S-Q4 (test sequences). The baseline localization without change detection is evaluated independently on all four query sequences. The results are presented in Table 6.10 and examples of matched places for query images from one place are shown in Figure 6.10.

Table 6.10: Localization accuracy of the proposed method on the query sequences from the STRANDS Witham Wharf data set using the SURF features. Better results are shown in bold.

Visual database	Updated on	Query sequence	Change detection	Localization accuracy
S-DB	–	S-Q1	off	64.15 %
S-DB	**–**	**S-Q1**	**on**	**66.37 %**
S-DB	–	S-Q2	off	58.68 %
S-DB	**S-Q1**	**S-Q2**	**on**	**60.16 %**
S-DB	–	S-Q3	off	60.76 %
S-DB	**S-Q1**	**S-Q3**	**on**	**62.67 %**
S-DB	–	S-Q4	off	47.83 %
S-DB	**S-Q1**	**S-Q4**	**on**	**50.61 %**

Note that the robot records the images 24 hours and therefore, a substantial portion of the images, which were taken at night, is completely dark. Such images do not contain any features and therefore cannot be matched correctly. To ensure a fair evaluation, we have not excluded any images from the data set.

The sequences are processed in batches of eight images, one from each place. Once the images from the same time frame are processed, the algorithm moves to the next time frame. If the change detection module is enabled, the weights are updated after processing every image. Already on the query sequence S-Q1, during ongoing change detection, a higher localization

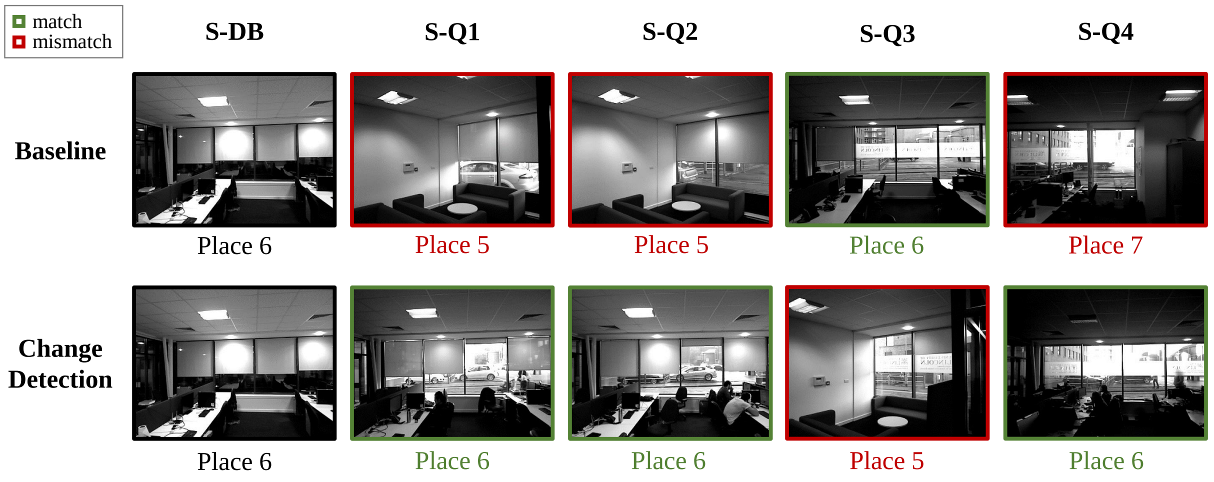


Figure 6.10: Examples of images matched by the baseline method and using the change detection. The figure shows successful matches (S-Q1, S-Q2, S-Q4) as well as a localization failure (S-Q3) when using the change detection method.

performance is achieved with the change detection module enabled. For the test sequences S-Q2, S-Q3, and S-Q4, feature weights pre-trained by processing S-Q1 are used as a starting point and they continue to update throughout the experiment. Also on all the test sequences, the localization is more accurate when the change detection module is enabled.

6.6.3. Adaptive Appearance-Based Map

An alternative to the proposed method is the Adaptive Appearance-Based Map (AABM) for long-term localization, presented in [36]. The algorithm introduces two types of feature storage: the short-term memory (STM) and the long-term memory (LTM). While LTM is used for feature matching, STM serves as a waiting list for newly discovered features captured at the same place. Both types of memory are adaptively updated based on the features observed in the query images. For a detailed description of the method, please refer to [36].

We evaluate AABM in the same way as our method (Section 6.6.2). To configure the method, we adopt the parameters used in the experiments reported in [36]. As the authors do not suggest how to tune the parameters for specific data sets and since both experiments presented in [36] share similar properties with the Witham Wharf data set, we have executed the evaluation in the two configurations used in the work. The first one uses STM with 4 stages and LTM with 5 stages, while the second one is configured with a 2-stage STM and a 4-stage LTM. We used the same baseline localization method as in our other experiments, which is similar to the *static map* described in [36]. The results are reported in Table 6.11.

To ensure a fair comparison, the STM and LTM update (denoted *rehearsal* and *recall* in [36]) was performed only if the number of matching features between the query image and the best-match LTM was at least $\theta = 10$, same as in our method. Without this modification, the LTM would be cleared after observing several successive fully dark images and the method would stop working.

Table 6.11: Localization accuracy of [36] on the query sequences from the STRANDS Witham Wharf data set. Performance better than our method (Table 6.10) is denoted in bold.

Base images	Pre-trained on	Query sequence	STM stages	LTM stages	Localization accuracy
S-DB	**–**	**S-Q1**	**2**	**4**	**66.47 %**
S-DB	–	S-Q1	4	5	55.50 %
S-DB	**S-Q1**	**S-Q2**	**2**	**4**	**61.89 %**
S-DB	S-Q1	S-Q2	4	5	55.38 %
S-DB	S-Q1	S-Q3	2	4	57.29 %
S-DB	S-Q1	S-Q3	4	5	49.05 %
S-DB	S-Q1	S-Q4	2	4	43.14 %
S-DB	S-Q1	S-Q4	4	5	45.05 %

First, we discuss the performance of AABM using 2 STM stages and 4 LTM stages. The method [36] shows a comparable to slightly better performance w.r.t. our method (Table 6.10) on images recorded in the period of the first eight days (0.10 % for S-Q1 and 1.73 % for S-Q2). However, the performance drops substantially for images captured three months later (5.38 % on S-Q3) and one year later (7.47 % on S-Q4). For the configuration of 4 STM stages and 5 LTM stages, our change detection method outperforms AABM on all query sequences.

The results indicate that the performance of AABM varies depending on the number of STM and LTM stages. The advantage of our method is that its performance is not particularly dependent on the choice of parameters – we did not have to change the configuration of the method in any of the performed experiments. Apart from the setting of the parameters, the difference in the performance can be explained by the way the features are processed. Our method accentuates the features from the initial scanning that remain the same and reduces the impact of those that have changed by updating their weights. By contrast, AABM completely drops LTM features that are not present for several frames and needs to re-learn them again, passing them first through STM. Therefore, feature weighting in our method provides a better way to capture feature importance.

6.6.4. Localization Using FAB-MAP

This section presents an evaluation of a general localization method that does not update the database. We chose the Fast Appearance-Based Mapping [35], abbreviated as FAB-MAP, which is a popular method for localization and mapping.

FAB-MAP is a probabilistic approach based on Chow Liu trees [33]. The method learns a generative model of bag-of-words observations [131]. The requirement for a learned model (*vocabulary*) represents the main difference from our method and AABM. However, FAB-MAP is expected to perform well even with a model trained on different data, which diminishes the issue with the training overhead. The authors show that the method is able to deal with scene changes, which is applicable to our experimental scenario.

We have used a publicly available implementation[1] of the FAB-MAP algorithm with the Indoor Environments vocabulary[2]. We have adapted several parameters in the default config-

[1]http://www.robots.ox.ac.uk/~mjc/Software.htm
[2]http://www.robots.ox.ac.uk/~mjc/FabMap_Release/IndoorVocab_10k.zip

uration of the method to match our evaluation scenario on the STRANDS Witham Wharf data set. The SURF blob response threshold was changed to 4.0, as suggested in the documentation of the vocabulary used. The prior probability of being at a new place was set to zero, because we require each image to be assigned to one of the existing locations, due to the nature of the Witham Wharf data set. Finally, the position prior model was set to uniform as the images in the Witham Wharf data set do not come from a continuous robot trajectory, but they are recorded at individual locations.

Same as in Sections 6.6.2 and 6.6.3, we have used S-DB as the base environment representation and S-Q1 to S-Q4 as the query sequences, see Table 6.9. Each query image was assigned to one of the eight locations in S-DB. The overall accuracy for each query sequence is reported in Table 6.12.

Table 6.12: Comparison of the localization accuracy of FAB-MAP [35] and the proposed change detection method on the query sequences from the STRANDS Witham Wharf data set. The better performing method is denoted in bold.

Base images	Query sequence	Method	Localization accuracy
S-DB	**S-Q1**	**FAB-MAP**	**67.42 %**
S-DB	S-Q1	Our method	66.37 %
S-DB	S-Q2	FAB-MAP	59.64 %
S-DB	**S-Q2**	**Our method**	**60.16 %**
S-DB	S-Q3	FAB-MAP	57.73 %
S-DB	**S-Q3**	**Our method**	**62.67 %**
S-DB	S-Q4	FAB-MAP	46.53 %
S-DB	**S-Q4**	**Our method**	**50.61 %**

The results show that FAB-MAP, being a more sophisticated technique, expectably outperforms our method, and also AABM, on the initial query sequence featuring a small number of changes. However, from the performance on the following sequences, we conclude that our method performs better on data recorded several months later after the initial scanning, similarly as in comparison with AABM discussed in Section 6.6.3. The proposed method with change detection is better by 4.94 % on S-Q3 and by 4.08 % on S-Q4 than FAB-MAP. Therefore, this again confirms the benefits of change detection for images recorded after a longer period of time has passed from the initial scanning, where more changes are present in the environment.

6.6.5. Feature Types

As discussed in Section 6.3.1, the proposed method can be used with various feature detectors and descriptors, according to the choice of the user. In this experiment, we compare SURF [14], ORB [123], and BRISK features [94]. In all cases, we use the same method for feature detection and for descriptor calculation.

The ORB and BRISK features have binary descriptors. In contrast to the SURF descriptors, which can be compared by using the Euclidean distance, these descriptors are compared through the Hamming distance [94, 123]. The ORB and BRISK descriptor distances are normalized to the range of the SURF descriptor distances for consistency.

We have evaluated the method in an identical scenario as in Section 6.6.2. The baseline algorithm without change detection is run independently on all four query sequences without any prior weight updates. The change detection is executed first on the sequence S-Q1 and then the updated visual database is used as an entry point to evaluate the sequences S-Q2, S-Q3, and S-Q4. The results are presented in Table 6.13 for the ORB features and in Table 6.14 for the BRISK features.

Table 6.13: Localization accuracy of the proposed method on the query sequences from the STRANDS Witham Wharf data set using the ORB features. Better results are shown in bold.

Visual database	Updated on	Query sequence	Change detection	Localization accuracy
S-DB	–	S-Q1	off	56.24 %
S-DB	**–**	**S-Q1**	**on**	**56.65 %**
S-DB	–	S-Q2	off	52.26 %
S-DB	**S-Q1**	**S-Q2**	**on**	**53.73 %**
S-DB	–	S-Q3	off	48.26 %
S-DB	**S-Q1**	**S-Q3**	**on**	**48.96 %**
S-DB	**–**	**S-Q4**	**off**	**37.59 %**
S-DB	S-Q1	S-Q4	on	37.50 %

Table 6.14: Localization accuracy of the proposed method on the query sequences from the STRANDS Witham Wharf data set using the BRISK features. Better results are shown in bold.

Visual database	Updated on	Query sequence	Change detection	Localization accuracy
S-DB	–	S-Q1	off	61.35 %
S-DB	**–**	**S-Q1**	**on**	**62.18 %**
S-DB	–	S-Q2	off	54.60 %
S-DB	**S-Q1**	**S-Q2**	**on**	**55.82 %**
S-DB	–	S-Q3	off	47.57 %
S-DB	**S-Q1**	**S-Q3**	**on**	**50.43 %**
S-DB	–	S-Q4	off	40.36 %
S-DB	**S-Q1**	**S-Q4**	**on**	**42.45 %**

The results show that an improvement with respect to the baseline is achieved by using the change detection method with alternative feature types in almost all cases, with the only exception of the query sequence S-Q4 evaluated using the ORB features. However, in that case, the localization accuracy is low, both without and with the change detection. The SURF features (Table 6.10) achieve the best localization accuracy on all query sequences. The average computation times per query image for implementation in Matlab using the Computer Vision Toolbox on a standard computer[3] are 163 ms for the SURF features, 363 ms for the ORB features, and 880 ms for the BRISK features. The algorithm therefore runs in the shortest time using the SURF features. Note that in all cases, we used the default parameters of the localization method, of the detector, and of the descriptor. We did not perform any parameter tuning.

[3]Intel Core i7-4610M @ 3.0 GHz, 16 GB RAM

6.6.6. DISCUSSION

The evaluation on the STRANDS Witham Wharf data set has demonstrated that the proposed method is able to achieve improved localization accuracy over the baseline and outperform the alternative methods [35, 36] in a long-term scenario. The performance of these methods could possibly be improved by extensive parameter tuning, but the authors do not provide any guidelines for setting the parameters. The strength of our method is therefore the small number of parameters and the robustness of the method to their setting, as demonstrated by using the same configuration for substantially different data sets.

The change detection method allows for the use of different feature types. SURF features yield the best results both in the localization accuracy and in the processing time.

The Witham Wharf data set contains images assigned to the same place taken from slightly different viewpoints, as well as under substantial day/night illumination changes. Such situations increase the risk of false matches and subsequent wrong feature updates. The experimental results show that the proposed method yields improved performance over the baseline despite these challenges.

6.7. CONCLUSIONS

We have proposed a method for change detection based on the comparison of local visual features and we have shown how the change detection method can be incorporated into a localization framework. The representation of the environment in the form of a visual database is continuously adapted as the robot moves through its area of operation. We have introduced feature weights to capture the importance of each feature. The weights are updated based on the feature descriptor similarity.

We have used a three-component confidence criterion to decrease the risk of impairing the visual database by introducing changes from wrongly matched images. The experimental evaluation on four different environments has shown that the change detection algorithm allows to capture the dynamics of the environment and leads to more accurate localization.

An evaluation on the Witham Wharf data set with thousands of images provided an insight into the method performance in additional experiments. The proposed method outperforms the baseline localization method without change detection and also two alternative approaches, AABM and FAB-MAP, on the majority of the query sequences.

In our future work, we are planning to perform a long-term experiment by recording multiple large-scale sequences over a long time span and to further improve the algorithm by evaluating other feature weight update methods. Using a precise localization based on data from additional sensors or from a motion capture system would improve the accuracy in the visual database building stage. This would reduce reprojection errors in image matching and allow for propagating the detected changes also to the neighboring records, improving the overall localization accuracy.

Another possible line of future research would be to incorporate the semantic information into the change detection algorithm, e.g., by applying an object detection method to determine the changes in the scenes based on object occurrence.

7
Conclusions and Future Research

7.1. Conclusions

The objective of this thesis was to address a number of challenges in data-driven model learning. To that end, efficient techniques have been proposed to construct an accurate model of a robot and its environment and use it for tasks such as robot control and localization.

Chapter 3 introduced symbolic regression as an effective tool for constructing robot models from data. SR produces accurate, compact models with a low number of parameters using even very small training sets. In contrast to black-box methods such as deep neural networks, it constructs models in the form of analytic expressions, allowing for an insight into the model structure. In terms of data efficiency, real-world experiments with an under-actuated pendulum have shown that only 5 seconds of interaction with the system are sufficient to find an accurate model that can be successfully used within an RL controller to swing up the pendulum. The experiments on a walking robot with 14-dimensional state space have shown that accurate models can be found by means of SR even for higher-dimensional systems while the computational cost and the resulting model complexity grow only linearly.

The approach to model learning proposed in this work is not dependent on a particular choice of the SR algorithm. It has been demonstrated that two distinct SR algorithms, SNGP and MGGP, yield comparable results. Next to constructing standard state-space models, SR has been used to build also input–output models. The advantage of input–output models is that they do not require observations of the full state vector, which is often not directly measurable in practice. A comparison with widely-used state-of-the-art methods – neural networks and local linear regression – has shown that SR performs on the inverted pendulum problem by at least one order of magnitude better than these alternatives in terms of RMSE on the test set. The findings presented in this thesis aim at achieving a broader adoption of SR as a viable alternative to established data-driven model learning techniques in the field of robotics and dynamic systems.

One of the challenges in data-driven model learning consists in large data sets collected during the robot operation. Large data sets not only represent an excessive computational burden for model learning, but they are also typically composed of a majority of samples from repetitive motions, resulting in an uneven data distribution. Using such data for model learning in turn produces a strong bias in the models. To that end, in Chapter 4, a novel method based on the model prediction error, called PERMIT, has been proposed and compared to four sample selection methods from the literature. The methods have been evaluated on data from three dynamic systems: a simulated mobile robot, a real mobile robot TurtleBot 2, and a real Parrot Bebop 2 drone. The results have shown that methods exploiting the information about the data or intermediate models clearly outperform sequential and random sample selection methods. The proposed method PERMIT ranked among the best-performing ones together with the variance method. An advantage of PERMIT over the variance method is that it does not require a set of models but only one model to select the training samples. In a real-world experiment with a mobile robot, a model found by SR using a training data set of as few as 24 samples was used within an RL framework to successfully perform the motion control task.

Data-driven model learning methods often produce models that are accurate, but do not comply with the physical constraints of the robot. Such models used in a model-based control framework may have an adverse impact on the resulting control policies. To deal with that, in Chapter 5, SR has been extended to include prior knowledge about the system, which is in most cases known. The prior knowledge can be represented in the form of prior models, such as theoretical or empirical models, and formal constraints. The prior knowledge enters the model search along with the training data with the objective to evolve accurate robot models more efficiently, using small training data sets, and following the formal constraints. A multi-objective approach is used to minimize the data fitting error as well as the constraint violation error at the same time. An experimental evaluation on two robots of different complexity has shown that including the prior knowledge improves the model accuracy and compliance with the physical properties of the robot, as compared to the baseline SR. An improvement in model accuracy has been achieved by including the prior model both for the baseline SNGP and for SNGP with formal constraints.

Chapter 6 presented a method for visual localization of mobile robots in dynamic environments. In contrast to deep neural networks, the proposed method based on local features represents a data-efficient and computationally lightweight solution. An experimental evaluation on our own long-term localization data set composed of four different environments has shown that using the change detection algorithm leads to more accurate localization. The localization results on the public Witham Wharf data set with thousands of images have shown that the proposed method outperforms not only the baseline localization method without change detection, but also two alternative state-of-the-art approaches, AABM and FAB-MAP, on a majority of the query sequences.

7.2. FUTURE RESEARCH

While solutions to several challenges in data-driven model learning have been proposed in this thesis, many interesting research questions remain open for future investigation. The models found by means of SR are compact, in particular when compared to deep neural networks. However, the model complexity can be controlled only indirectly through user-defined parameters such as the maximum number of features, maximum feature depth, and the set of elementary functions. To have direct control over the accuracy-complexity tradeoff, a future line of research would investigate methods for progressive model construction and reduction.

The sample selection methods presented in Chapter 4 are used for SR, but they could also be applied to other data-driven model learning approaches, such as local linear regression [10]. Another possible application would be to control the composition of the replay buffer in deep reinforcement learning. The next step in further developing the sample selection framework would be to introduce an automated data set maintenance, including removal of data samples that are either outliers or are no longer valid because the robot model has changed over time.

This thesis deals with using SR for robot model learning and using the resulting model within an RL framework. However, SR can also be used, for instance, to model the value function in RL, as in [90]. The value functions in that work are constructed using baseline SNGP. Therefore, a line of future research would be to apply the proposed SR extensions, in particular including the prior knowledge and using the sample selection, to construct value functions.

The accuracy of robot localization in dynamic environments using the proposed change detection method could be improved by reducing reprojection errors in image matching. Possible solutions include using stereo images or combining RGB images with depth data, motivated by the increasing availability of RGB-D sensors. A more accurate matching would improve the precision of weight updates and it would also allow propagating the detected changes to the neighboring records. An interesting extension to using local features within the change detection algorithm would be to capture the semantic information in the scene, for instance, through object detection. Then, the tracked changes in the scenes would include object occurrences, which could lead to higher robustness and reliability of the localization algorithm.

Grants

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.

List of Acronyms

AABM	Adaptive Appearance-Based Map
AI	Artificial Intelligence
BRIEF	Binary Robust Independent Elementary Features
BRISK	Binary Robust Invariant Scalable Keypoint
DNN	Deep Neural Network
DOF	Degree Of Freedom
ELU	Exponential Linear Unit
FAB-MAP	Fast Appearance-Based MAPping
FAST	Features from Accelerated Segment Test
FSH	Feature Stability Histogram
GMM	Gaussian Mixture Model
GP	Genetic Programming
HOG	Histogram of Oriented Gradients
LiDAR	Light Detection And Ranging
LLR	Local Linear Regression
LTM	Long-Term Memory
MGGP	Multi-Gene Genetic Programming
NARX	Nonlinear AutoRegressive with eXogenous input
NSGA	Non-dominated Sorting Genetic Algorithm
ORB	Oriented FAST and Rotated BRIEF
PERMIT	Prediction ERror method for Model ImprovemenT
RANSAC	RANdom SAmple Consensus
ReLU	Rectified Linear Unit
RGB	Red, Green, Blue
RGB-D	Red, Green, Blue, Depth
RL	Reinforcement Learning
RMS	Root Mean Square
RMSE	Root Mean Square Error

SGD	Stochastic Gradient Descent
SIFT	Scale-Invariant Feature Transform
SLAM	Simultaneous Localization And Mapping
SNGP	Single Node Genetic Programming
SR	Symbolic Regression
STM	Short-Term Memory
SURF	Speeded-Up Robust Features
TSDF	Truncated Signed Distance Function

Bibliography

[1] I. Abraham and T. D. Murphey, *Active learning of dynamics for data-driven control using Koopman operators*, IEEE Transactions on Robotics, 35 (2019), pp. 1071–1083.

[2] P. F. Alcantarilla, S. Stent, G. Ros, R. Arroyo, and R. Gherardi, *Street-view change detection with deconvolutional networks*, Autonomous Robots, 42 (2018), pp. 1301–1322.

[3] E. Alibekov, J. Kubalík, and R. Babuška, *Symbolic method for deriving policy in reinforcement learning*, in 2016 IEEE 55th Conference on Decision and Control (CDC), Dec 2016, pp. 2789–2795.

[4] R. Alterovitz, S. Koenig, and M. Likhachev, *Robot planning in the real world: Research challenges and opportunities*, AI Magazine, 37 (2016), pp. 76–84.

[5] J. Andrade-Cetto and A. Sanfeliu, *Concurrent map building and localization on indoor dynamic environments*, International Journal of Pattern Recognition and Artificial Intelligence, 16 (2002), pp. 361–374.

[6] M. H. Ang, O. Khatib, and B. Siciliano, *Encyclopedia of Robotics*, Springer, 2018.

[7] 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 (GECCO '14), 2014, pp. 879–886.

[8] I. Arnaldo, U.-M. O'Reilly, and K. Veeramachaneni, *Building predictive models via feature synthesis*, in Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation (GECCO '15), 2015, pp. 983–990.

[9] R. Arroyo, P. F. Alcantarilla, L. M. Bergasa, and E. Romera, *Towards life-long visual localization using an efficient matching of binary sequences from images*, in 2015 IEEE International Conference on Robotics and Automation (ICRA), 2015, pp. 6328–6335.

[10] C. G. Atkeson, A. W. Moore, and S. Schaal, *Locally weighted learning*, Artificial Intelligence Review, 11 (1997), pp. 11–73.

[11] R. C. Atkinson and R. M. Shiffrin, *Human memory: A proposed system and its control processes*, vol. 2 of Psychology of Learning and Motivation, Academic Press, 1968, pp. 89–195.

[12] B. Bacca, J. Salvi, and X. Cufí, *Appearance-based mapping and localization for mobile robots using a feature stability histogram*, Robotics and Autonomous Systems, 59 (2011), pp. 840–857.

[13] E. BARKER AND C. RAS, *Unsupervised basis function adaptation for reinforcement learning*, Journal of Machine Learning Research, 20 (2019), pp. 1–73.

[14] H. BAY, A. ESS, T. TUYTELAARS, AND L. VAN GOOL, *Speeded-up robust features (SURF)*, Computer Vision and Image Understanding, 110 (2008), pp. 346–359.

[15] B. BESCÓS, J. M. FÁCIL, J. CIVERA, AND J. NEIRA, *DynSLAM: Tracking, mapping and inpainting in dynamic scenes*, arXiv preprint, abs/1806.05620 (2018).

[16] J. C. BEZDEK, *Pattern recognition with fuzzy objective function algorithms*, Springer Science & Business Media, 2013.

[17] J. BISWAS AND M. VELOSO, *Episodic non-Markov localization: Reasoning about short-term and long-term features*, in 2014 IEEE International Conference on Robotics and Automation (ICRA), 2014, pp. 3969–3974.

[18] I. BŁĄDEK AND K. KRAWIEC, *Solving symbolic regression problems with formal constraints*, in Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '19), 2019, pp. 977–984.

[19] J. BOEDECKER, J. T. SPRINGENBERG, J. WÜLFING, AND M. RIEDMILLER, *Approximate real-time optimal control based on sparse Gaussian process models*, in 2014 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL), Dec 2014, pp. 1–8.

[20] J. BONGARD AND H. LIPSON, *Automated reverse engineering of nonlinear dynamical systems*, Proceedings of the National Academy of Sciences, 104 (2007), pp. 9943–9948.

[21] F. BONIARDI, T. CASELITZ, R. KÜMMERLE, AND W. BURGARD, *A pose graph-based localization system for long-term navigation in CAD floor plans*, Robotics and Autonomous Systems, 112 (2019), pp. 84–97.

[22] N. BORE, P. JENSFELT, AND J. FOLKESSON, *Multiple object detection, tracking and long-term dynamics learning in large 3D maps*, arXiv preprint, abs/1801.09292 (2018).

[23] J. BRANKE, S. GRECO, R. SŁOWIŃSKI, AND P. ZIELNIEWICZ, *Learning value functions in interactive evolutionary multiobjective optimization*, IEEE Transactions on Evolutionary Computation, 19 (2015), pp. 88–102.

[24] C. BRAUER, *Using Eureqa in a stock day-trading application*. Cypress Point Technologies, LLC, 2012.

[25] 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), 2019, pp. 6244–6250.

[26] L. BUŞONIU, D. ERNST, R. BABUŠKA, AND B. DE SCHUTTER, *Approximate dynamic programming with a fuzzy parameterization*, Automatica, 46 (2010), pp. 804–814.

[27] L. BUŞONIU, D. ERNST, B. DE SCHUTTER, AND R. BABUŠKA, *Cross-entropy optimization of control policies with adaptive basis functions*, IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics, 41 (2011), pp. 196–209.

[28] R. BURBIDGE, J. J. ROWLAND, AND R. D. KING, *Active learning for regression based on query by committee*, in International Conference on Intelligent Data Engineering and Automated Learning, Springer, 2007, pp. 209–218.

[29] J. C. BUTCHER AND N. GOODWIN, *Numerical methods for ordinary differential equations*, vol. 2, Wiley Online Library, 2008.

[30] W. CAARLS, *Generic Reinforcement Learning Library*, 2018. https://github.com/wcaarls/grl.

[31] N. CARLEVARIS-BIANCO AND R. M. EUSTICE, *Learning visual feature descriptors for dynamic lighting conditions*, in 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2014, pp. 2769–2776.

[32] Z. CHEN, L. LIU, I. SA, Z. GE, AND M. CHLI, *Learning context flexible attention model for long-term visual place recognition*, IEEE Robotics and Automation Letters, 3 (2018), pp. 4015–4022.

[33] C. CHOW AND C. LIU, *Approximating discrete probability distributions with dependence trees*, IEEE Transactions on Information Theory, 14 (1968), pp. 462–467.

[34] W. CHURCHILL AND P. NEWMAN, *Experience-based navigation for long-term localisation*, The International Journal of Robotics Research, 32 (2013), pp. 1645–1661.

[35] M. CUMMINS AND P. NEWMAN, *FAB-MAP: Probabilistic localization and mapping in the space of appearance*, The International Journal of Robotics Research, 27 (2008), pp. 647–665.

[36] F. DAYOUB AND T. DUCKETT, *An adaptive appearance-based map for long-term topological localization of mobile robots*, in 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2008, pp. 3364–3369.

[37] T. DE BRUIN, J. KOBER, K. TUYLS, AND R. BABUŠKA, *Experience selection in deep reinforcement learning for control*, Journal of Machine Learning Research, 19 (2018), pp. 1–56.

[38] T. DE BRUIN, J. KOBER, K. TUYLS, AND R. BABUŠKA, *Integrating state representation learning into deep reinforcement learning*, IEEE Robotics and Automation Letters, 3 (2018), pp. 1394–1401.

[39] K. DEB, A. PRATAP, S. AGARWAL, AND T. MEYARIVAN, *A fast and elitist multiobjective genetic algorithm: NSGA-II*, IEEE Transactions on Evolutionary Computation, 6 (2002), pp. 182–197.

[40] M. DEISENROTH AND C. E. RASMUSSEN, *PILCO: A model-based and data-efficient approach to policy search*, in International Conference on Machine Learning (ICML), 2011, pp. 465–472.

[41] M. P. DEISENROTH, *Efficient reinforcement learning using Gaussian processes*, KIT Scientific Publishing, 2010.

[42] E. DERNER, C. GOMEZ, A. C. HERNANDEZ, R. BARBER, AND R. BABUŠKA, *Change detection using weighted features for image-based localization*, Robotics and Autonomous Systems, 135 (2021), p. 103676.

[43] E. DERNER, J. KUBALÍK, N. ANCONA, AND R. BABUŠKA, *Constructing parsimonious analytic models for dynamic systems via symbolic regression*, Applied Soft Computing, 94 (2020), p. 106432.

[44] 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. 5105–5112.

[45] E. DERNER, J. KUBALÍK, AND R. BABUŠKA, *Reinforcement learning with symbolic input-output models*, in 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Oct 2018, pp. 3004–3009.

[46] E. DERNER, J. KUBALÍK, AND R. BABUŠKA, *Guiding robot model construction with prior features*, in 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Oct 2021, pp. 7112–7118.

[47] E. DERNER, J. KUBALÍK, AND R. BABUŠKA, *Selecting informative data samples for model learning through symbolic regression*, IEEE Access, 9 (2021), pp. 14148–14158.

[48] P. DREWS, L. J. MANSO, S. DA SILVA FILHO, AND P. NÚÑEZ, *Improving change detection using Vertical Surface Normal Histograms and Gaussian Mixture Models in structured environments*, in 16th International Conference on Advanced Robotics (ICAR), 2013, pp. 1–7.

[49] O. EGUASA, E. EDIONWE, AND J. MBEGBU, *Local linear regression and the problem of dimensionality: A remedial strategy via a new locally adaptive bandwidths selector*, Journal of Applied Statistics, (2022), pp. 1–27.

[50] D. ERNST, P. GEURTS, AND L. WEHENKEL, *Tree-based batch mode reinforcement learning*, Journal of Machine Learning Research, 6 (2005), pp. 503–556.

[51] S. ERTEKIN, J. HUANG, L. BOTTOU, AND L. GILES, *Learning on the border: Active learning in imbalanced data classification*, in Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management, 2007, pp. 127–136.

[52] J. FAIGL, J. CHUDOBA, K. KOŠNAR, M. KULICH, M. SASKA, AND L. PŘEUČIL, *SyRoTek – A robotic system for education*, AT&P Journal PLUS, 2 (2010), pp. 31–36.

[53] M. FEHR, F. FURRER, I. DRYANOVSKI, J. STURM, I. GILITSCHENSKI, R. SIEGWART, AND C. CADENA, *TSDF-based change detection for consistent long-term dense reconstruction and dynamic object discovery*, in 2017 IEEE International Conference on Robotics and Automation (ICRA), 2017, pp. 5237–5244.

[54] M. L. FELIS, *RBDL: An efficient rigid-body dynamics library using recursive algorithms*, Autonomous Robots, 41 (2017), pp. 495–511.

[55] C. FERREIRA, *Gene expression programming: A new adaptive algorithm for solving problems*, Complex Systems, 13 (2001), pp. 87–129.

[56] R. FINMAN, T. WHELAN, M. KAESS, AND J. J. LEONARD, *Toward lifelong object segmentation from change detection in dense RGB-D maps*, in 2013 European Conference on Mobile Robots, 2013, pp. 178–185.

[57] M. A. FISCHLER AND R. C. BOLLES, *Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography*, Communications of the ACM, 24 (1981), pp. 381–395.

[58] J. FORBES AND D. ANDRE, *Representations for learning control policies*, in Proceedings of the ICML-2002 Workshop on Development of Representations, 2002, pp. 7–14.

[59] E. GARCIA-FIDALGO AND A. ORTIZ, *Vision-based topological mapping and localization methods: A survey*, Robotics and Autonomous Systems, 64 (2015), pp. 1–20.

[60] D. GEDON, N. WAHLSTRÖM, T. B. SCHÖN, AND L. LJUNG, *Deep state space models for nonlinear system identification*, in 19th IFAC Symposium on System Identification (SYSID), vol. 54, Elsevier BV, 2021, pp. 481–486.

[61] J. GERTLER, *Fault detection and diagnosis*, Springer, 2013.

[62] J. D. GIBBONS AND S. CHAKRABORTI, *Nonparametric statistical inference: Revised and expanded*, CRC Press, 2014.

[63] S. GREYDANUS, M. DZAMBA, AND J. YOSINSKI, *Hamiltonian neural networks*, in Advances in Neural Information Processing Systems, 2019, pp. 15379–15389.

[64] I. GRONDMAN, M. VAANDRAGER, L. BUŞONIU, R. BABUŠKA, AND E. SCHUITEMA, *Efficient model learning methods for actor–critic control*, IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics, 42 (2012), pp. 591–602.

[65] S. GU, T. P. LILLICRAP, I. SUTSKEVER, AND S. LEVINE, *Continuous deep Q-learning with model-based acceleration*, arXiv preprint, abs/1603.00748 (2016).

[66] N. HAWES, C. BURBRIDGE, F. JOVAN, L. KUNZE, B. LACERDA, L. MUDROVA, J. YOUNG, J. WYATT, D. HEBESBERGER, T. KORTNER, ET AL., *The STRANDS project: Long-term autonomy in everyday environments*, IEEE Robotics & Automation Magazine, 24 (2017), pp. 146–156.

[67] N. HEESS, G. WAYNE, D. SILVER, T. LILLICRAP, T. EREZ, AND Y. TASSA, *Learning continuous control policies by stochastic value gradients*, in Advances in Neural Information Processing Systems, 2015, pp. 2944–2952.

[68] T. HESTER AND P. STONE, *Intrinsically motivated model learning for developing curious robots*, Artificial Intelligence, 247 (2017), pp. 170–186.

[69] H. O. İLHAN AND M. F. AMASYAL, *Comparing informative sample selection strategies in classification ensembles*, International Journal of Machine Learning and Computing, 4 (2014), p. 79.

[70] D. JACKSON, *A new, node-focused model for genetic programming*, in Genetic Programming: 15th European Conference, EuroGP 2012 Proceedings, A. Moraglio, S. Silva, K. Krawiec, P. Machado, and C. Cotta, eds., 2012, pp. 49–60.

[71] D. JACKSON, *Single node genetic programming on problems with side effects*, in Parallel Problem Solving from Nature – PPSN XII: 12th International Conference Proceedings, Part I, C. A. C. Coello, V. Cutello, K. Deb, S. Forrest, G. Nicosia, and M. Pavone, eds., 2012, pp. 327–336.

[72] T. A. JOHANSEN AND B. A. FOSS, *ORBIT – operating-regime-based modeling and identification toolkit*, Control Engineering Practice, 6 (1998), pp. 1277–1286.

[73] E. JOHNS AND G.-Z. YANG, *Feature co-occurrence maps: Appearance-based localisation throughout the day*, in 2013 IEEE International Conference on Robotics and Automation (ICRA), 2013, pp. 3212–3218.

[74] E. JOHNS AND G.-Z. YANG, *Generative methods for long-term place recognition in dynamic scenes*, International Journal of Computer Vision, 106 (2014), pp. 297–314.

[75] N. K. JONG AND P. STONE, *Model-based function approximation in reinforcement learning*, in Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS '07, 2007, pp. 95:1–95:8.

[76] 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 Proceedings, 2019, pp. 2673–2680.

[77] N. KHOSHNEVIS AND R. TABORDA, *Application of pool-based active learning in physics-based earthquake ground-motion simulation*, Seismological Research Letters, 90 (2019), pp. 614–622.

[78] J. KIEFER AND J. WOLFOWITZ, *Stochastic estimation of the maximum of a regression function*, The Annals of Mathematical Statistics, 23 (1952), pp. 462–466.

[79] D. P. KINGMA AND J. BA, *Adam: A method for stochastic optimization*, arXiv preprint, abs/1412.6980 (2014).

[80] J. KOBER AND J. PETERS, *Reinforcement learning in robotics: A survey*, in Reinforcement Learning: State-of-the-Art, M. Wiering and M. van Otterlo, eds., Springer, Berlin, Heidelberg, 2012, pp. 579–610.

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

[82] K. KONOLIGE AND J. BOWMAN, *Towards lifelong visual maps*, in 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2009, pp. 1156–1163.

[83] I. KORYAKOVSKIY, H. VALLERY, R. BABUŠKA, AND W. CAARLS, *Evaluation of physical damage associated with action selection strategies in reinforcement learning*, IFAC-PapersOnLine, 50 (2017), pp. 6928–6933.

[84] J. KOZA, *Genetic programming: On the programming of computers by means of natural selection (Complex adaptive systems)*, MIT Press Ltd, 1992.

[85] T. KRAJNÍK, J. P. FENTANES, O. M. MOZOS, T. DUCKETT, J. EKEKRANTZ, AND M. HANHEIDE, *Long-term topological localization for service robots in dynamic environments using spectral maps*, in 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2014.

[86] J. KUBALÍK, E. ALIBEKOV, AND R. BABUŠKA, *Optimal control via reinforcement learning with symbolic policy approximation*, IFAC-PapersOnLine, 50 (2017), pp. 4162–4167.

[87] 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.

[88] 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), 2020, pp. 958–966.

[89] J. KUBALÍK, E. DERNER, AND R. BABUŠKA, *Multi-objective symbolic regression for physics-aware dynamic modeling*, Expert Systems with Applications, 9 (2021), p. 115210.

[90] J. KUBALÍK, E. DERNER, J. ŽEGKLITZ, AND R. BABUŠKA, *Symbolic regression methods for reinforcement learning*, IEEE Access, 9 (2021), pp. 139697–139711.

[91] L. KUNZE, H. KARAOGUZ, J. YOUNG, F. JOVAN, J. FOLKESSON, P. JENSFELT, AND N. HAWES, *SOMA: A framework for understanding change in everyday environments using semantic object maps*, Association for the Advancement of Artificial Intelligence (AAAI), 2018.

[92] L. KUVAYEV AND R. S. SUTTON, *Model-based reinforcement learning with an approximate, learned model*, in Proceedings of the Ninth Yale Workshop on Adaptive and Learning Systems, 1996, pp. 101–105.

[93] S. LANGE, M. RIEDMILLER, AND A. VOIGTLANDER, *Autonomous reinforcement learning on raw visual input data in a real world application*, in Proceedings 2012 International Joint Conference on Neural Networks (IJCNN), Brisbane, Australia, June 2012, pp. 1–8.

[94] S. LEUTENEGGER, M. CHLI, AND R. Y. SIEGWART, *BRISK: Binary robust invariant scalable keypoints*, in 2011 International Conference on Computer Vision, 2011, pp. 2548–2555.

[95] S. LEVINE AND P. ABBEEL, *Learning neural network policies with guided policy search under unknown dynamics*, in Advances in Neural Information Processing Systems 27, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, eds., Curran Associates, Inc., 2014, pp. 1071–1079.

[96] R. LIECK AND M. TOUSSAINT, *Temporally extended features in model-based reinforcement learning with partial observability*, Neurocomputing, 192 (2016), pp. 49–60.

[97] T. P. LILLICRAP, J. J. HUNT, A. PRITZEL, N. HEESS, T. EREZ, Y. TASSA, D. SILVER, AND D. WIERSTRA, *Continuous control with deep reinforcement learning*, arXiv preprint, abs/1509.02971 (2015).

[98] R. LIOUTIKOV, A. PARASCHOS, J. PETERS, AND G. NEUMANN, *Sample-based information-theoretic stochastic optimal control*, in 2014 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2014, pp. 3896–3902.

[99] L. LJUNG, *System Identification: Theory for the user, second edition*, Prentice-Hall, New Jersey, 1999.

[100] D. G. LOWE, *Distinctive image features from scale-invariant keypoints*, International Journal of Computer Vision, 60 (2004), pp. 91–110.

[101] S. LOWRY, N. SÜNDERHAUF, P. NEWMAN, J. J. LEONARD, D. COX, P. CORKE, AND M. J. MILFORD, *Visual place recognition: A survey*, IEEE Transactions on Robotics, 32 (2015), pp. 1–19.

[102] T. MASTERS, *Neural, novel and hybrid algorithms for time series prediction*, John Wiley & Sons, Inc., 1995.

[103] 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), 2016, pp. 6500–6505.

[104] D. Q. MAYNE, J. B. RAWLINGS, C. V. RAO, AND P. O. SCOKAERT, *Constrained model predictive control: Stability and optimality*, Automatica, 36 (2000), pp. 789–814.

[105] J. F. MILLER AND P. THOMSON, *Cartesian genetic programming*, in Genetic Programming: European Conference, EuroGP Proceedings, R. Poli, W. Banzhaf, W. B. Langdon, J. Miller, P. Nordin, and T. C. Fogarty, eds., 2000, pp. 121–132.

[106] V. MNIH, K. KAVUKCUOGLU, D. SILVER, A. GRAVES, I. ANTONOGLOU, D. WIERSTRA, AND M. RIEDMILLER, *Playing Atari with deep reinforcement learning*, arXiv preprint, abs/1312.5602 (2013).

[107] 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, 518 (2015), pp. 529–533.

[108] R. MUNOS AND A. MOORE, *Variable resolution discretization in optimal control*, Machine Learning, 49 (2002), pp. 291–323.

[109] R. MURRAY-SMITH AND T. A. JOHANSEN, eds., *Multiple Model Approaches to Nonlinear Modeling and Control*, Taylor & Francis, London, UK, 1997.

[110] T. NASEER, W. BURGARD, AND C. STACHNISS, *Robust visual localization across seasons*, IEEE Transactions on Robotics, 34 (2018), pp. 289–302.

[111] B. NEUMAN, B. SOFMAN, A. STENTZ, AND J. A. BAGNELL, *Segmentation-based online change detection for mobile robots*, in 2011 IEEE International Conference on Robotics and Automation (ICRA), 2011, pp. 5427–5434.

[112] C. P. NEUMAN AND P. K. KHOSLA, *Identification of robot dynamics: An application of recursive estimation*, Springer US, Boston, MA, 1986, pp. 175–194.

[113] A. Y. NG, A. COATES, M. DIEL, V. GANAPATHI, J. SCHULTE, B. TSE, E. BERGER, AND E. LIANG, *Autonomous inverted helicopter flight via reinforcement learning*, in Experimental Robotics IX: The 9th International Symposium on Experimental Robotics, M. H. Ang and O. Khatib, eds., Springer, Berlin, Heidelberg, 2006, pp. 363–372.

[114] D. NGUYEN-TUONG AND J. PETERS, *Model learning for robot control: A survey*, Cognitive Processing, 12 (2011), pp. 319–340.

[115] D. NGUYEN-TUONG, M. SEEGER, AND J. PETERS, *Model learning with local Gaussian process regression*, Advanced Robotics, 23 (2009), pp. 2015–2034.

[116] D. NISTER AND H. STEWENIUS, *Scalable recognition with a vocabulary tree*, in 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06), vol. 2, 2006, pp. 2161–2168.

[117] F. NOBRE, C. HECKMAN, P. OZOG, R. W. WOLCOTT, AND J. M. WALLS, *Online probabilistic change detection in feature-based maps*, in 2018 IEEE International Conference on Robotics and Automation (ICRA), 2018, pp. 1–9.

[118] M. ONDERWATER, S. BHULAI, AND R. VAN DER MEI, *Value function discovery in Markov decision processes with evolutionary algorithms*, IEEE Transactions on Systems, Man, and Cybernetics: Systems, 46 (2016), pp. 1190–1201.

[119] J. PETERS AND S. SCHAAL, *Policy gradient methods for robotics*, in 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Oct 2006, pp. 2219–2225.

[120] A. PUNJANI AND P. ABBEEL, *Deep learning helicopter dynamics models*, in 2015 IEEE International Conference on Robotics and Automation (ICRA), 2015, pp. 3223–3230.

[121] J. B. RAWLINGS AND D. Q. MAYNE, *Model predictive control: Theory and design*, Nob Hill Publishing, 2009.

[122] N. ROY, I. POSNER, T. D. BARFOOT, P. BEAUDOIN, Y. BENGIO, J. BOHG, O. BROCK, I. DEPATIE, D. FOX, D. E. KODITSCHEK, T. LOZANO-PÉREZ, V. MANSINGHKA, C. J. PAL, B. RICHARDS, D. SADIGH, S. SCHAAL, G. S. SUKHATME, D. THÉRIEN, M. TOUSSAINT, AND M. VAN DE PANNE, *From machine learning to robotics: Challenges and opportunities for embodied intelligence*, arXiv preprint, abs/2110.15245 (2021).

[123] E. RUBLEE, V. RABAUD, K. KONOLIGE, AND G. BRADSKI, *ORB: An efficient alternative to SIFT or SURF*, in 2011 International Conference on Computer Vision, 2011, pp. 2564–2571.

[124] C. RYAN, J. COLLINS, AND M. O. NEILL, *Grammatical evolution: Evolving programs for an arbitrary language*, in Genetic Programming: First European Workshop, EuroGP 1998 Proceedings, W. Banzhaf, R. Poli, M. Schoenauer, and T. C. Fogarty, eds., 1998, pp. 83–96.

[125] M. SCHMIDT AND H. LIPSON, *Distilling free-form natural laws from experimental data*, Science, 324 (2009), pp. 81–85.

[126] E. SCHUITEMA, M. WISSE, T. RAMAKERS, AND P. JONKER, *The design of LEO: A 2D bipedal walking robot for online autonomous reinforcement learning*, in 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2010, pp. 3238–3243.

[127] D. P. SEARSON, *GPTIPS 2: An open-source software platform for symbolic data mining*, in Handbook of Genetic Programming Applications, Springer International Publishing, 2015, pp. 551–573.

[128] B. SETTLES, *Active learning literature survey*, tech. rep., University of Wisconsin-Madison Department of Computer Sciences, 2009.

[129] R. SIEGWART, I. R. NOURBAKHSH, AND D. SCARAMUZZA, *Introduction to autonomous mobile robots*, MIT Press, 2011.

[130] O. SIGAUD, C. SALAÜN, AND V. PADOIS, *On-line regression algorithms for learning mechanical models of robots: A survey*, Robotics and Autonomous Systems, 59 (2011), pp. 1115–1129.

[131] SIVIC AND ZISSERMAN, *Video Google: A text retrieval approach to object matching in videos*, in Proceedings of the Ninth IEEE International Conference on Computer Vision, 2003, pp. 1470–1477.

[132] F. SMARRA, G. D. DI GIROLAMO, V. DE IULIIS, A. JAIN, R. MANGHARAM, AND A. D'INNOCENZO, *Data-driven switching modeling for MPC using regression trees and random forests*, Nonlinear Analysis: Hybrid Systems, 36 (2020), p. 100882.

[133] N. STAELENS, D. DESCHRIJVER, E. VLADISLAVLEVA, B. VERMEULEN, T. DHAENE, AND P. DEMEESTER, *Constructing a no-reference H. 264/AVC bitstream-based video quality metric using genetic programming-based symbolic regression*, IEEE Transactions on Circuits and Systems for Video Technology, 99 (2012), pp. 1–12.

[134] F. STULP AND O. SIGAUD, *Many regression algorithms, one unified model: A review*, Neural Networks, 69 (2015), pp. 60–79.

[135] L. SUN, Z. YAN, A. ZAGANIDIS, C. ZHAO, AND T. DUCKETT, *Recurrent-OctoMap: Learning state-based map refinement for long-term semantic mapping with 3-D-Lidar data*, IEEE Robotics and Automation Letters, 3 (2018), pp. 3749–3756.

[136] R. S. SUTTON, *Dyna, an integrated architecture for learning, planning, and reacting*, SIGART Bulletin, 2 (1991), pp. 160–163.

[137] R. S. SUTTON AND A. G. BARTO, *Reinforcement learning: An introduction*, MIT Press, 2018.

[138] P. TORR AND A. ZISSERMAN, *MLESAC: A new robust estimator with application to estimating image geometry*, Computer Vision and Image Understanding, 78 (2000), pp. 138–156.

[139] H. J. TULLEKEN, *Generalized binary noise test-signal concept for improved identification-experiment design*, Automatica, 26 (1990), pp. 37–49.

[140] S.-M. UDRESCU AND M. TEGMARK, *AI Feynman: A physics-inspired method for symbolic regression*, Science Advances, 6 (2020), p. eaay2631.

[141] C. VALGREN AND A. J. LILIENTHAL, *SIFT, SURF and seasons: Long-term outdoor localization using local features*, in 3rd European Conference on Mobile Robots, 2007, pp. 253–258.

[142] V. VENKATASUBRAMANIAN, R. RENGASWAMY, K. YIN, AND S. N. KAVURI, *A review of process fault detection and diagnosis: Part I: Quantitative model-based methods*, Computers & Chemical Engineering, 27 (2003), pp. 293–311.

[143] 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.

[144] 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, 50 (2013), pp. 236–243.

[145] J. ŽEGKLITZ AND P. POŠÍK, *Linear combinations of features as leaf nodes in symbolic regression*, in Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO '17), 2017, pp. 145–146.

[146] K. WANG, G. PLEISS, J. GARDNER, S. TYREE, K. Q. WEINBERGER, AND A. G. WILSON, *Exact Gaussian processes on a million data points*, Advances in Neural Information Processing Systems, 32 (2019).

[147] R. WANG, R. GOYAL, S. CHAKRAVORTY, AND R. E. SKELTON, *Data-based control of partially-observed robotic systems*, in 2021 IEEE International Conference on Robotics and Automation (ICRA), 2021, pp. 8104–8110.

[148] L. WELLHAUSEN, R. DUBÉ, A. GAWEL, R. SIEGWART, AND C. CADENA, *Reliable real-time change detection and mapping for 3D LiDARs*, in 2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR), 2017, pp. 81–87.

[149] D. WU, C.-T. LIN, AND J. HUANG, *Active learning for regression using greedy sampling*, Information Sciences, 474 (2019), pp. 90–105.

[150] J. YAO, Y. WU, AND H. ZHAI, *Speeding up quantum few-body calculation with active learning*, arXiv preprint, abs/1904.10692 (2019).

[151] G. YEHUDA, M. GABEL, AND A. SCHUSTER, *It's not what machines can learn, it's what we cannot teach*, arXiv preprint, abs/2002.09398 (2020).

List of Author's Publications

This list of publications features all papers co-authored by the thesis author published before February 2022. All listed journal papers are indexed by Scopus and by the Web of Science.

Publications Related to the Thesis Topic

This section lists publications that are directly related to the thesis topic and cited in the thesis.

Publications in Impacted Journals

- E. Derner, J. Kubalík, and R. Babuška, *Selecting informative data samples for model learning through symbolic regression*, IEEE Access, 9 (2021), pp. 14148–14158.
  Citations: 2 (Google Scholar)

- E. Derner, C. Gomez, A. C. Hernandez, R. Barber, and R. Babuška, *Change detection using weighted features for image-based localization*, Robotics and Autonomous Systems, 135 (2021), p. 103676.
  Citations: 1 (Google Scholar) / 1 (Scopus)

- J. Kubalík, E. Derner, and R. Babuška, *Multi-objective symbolic regression for physics-aware dynamic modeling*, Expert Systems with Applications, 182 (2021), p. 115210.
  Citations: 1 (Google Scholar) / 1 (Scopus)

- J. Kubalík, E. Derner, J. Žegklitz, and R. Babuška, *Symbolic regression methods for reinforcement learning*, IEEE Access, 9 (2021), pp. 139697–139711.
  Citations: 7 (Google Scholar)

- E. Derner, J. Kubalík, N. Ancona, and R. Babuška, *Constructing parsimonious analytic models for dynamic systems via symbolic regression*, Applied Soft Computing, 94 (2020), p. 106432.
  Citations: 15 (Google Scholar) / 10 (Scopus)

Conference Publications

Conference publications indexed by Scopus and by the Web of Science:

- 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), 2020, pp. 958–966.
  Citations: 7 (Google Scholar) / 4 (Scopus)

- E. DERNER, C. GOMEZ, A. C. HERNANDEZ, R. BARBER, AND R. BABUŠKA, *Towards lifelong autonomy of mobile robots through feature-based change detection*, in 2019 European Conference on Mobile Robots (ECMR), Sep 2019, pp. 1–6.
  Citations: 1 (Google Scholar)

- E. DERNER, J. KUBALÍK, AND R. BABUŠKA, *Reinforcement learning with symbolic input-output models*, in 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Oct 2018, pp. 3004–3009.
  Citations: 11 (Google Scholar) / 6 (Scopus)

- 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. 5105–5112.
  Citations: 8 (Google Scholar) / 5 (Scopus)

Conference publications indexed by Scopus (to be indexed by the Web of Science):

- E. DERNER, J. KUBALÍK, AND R. BABUŠKA, *Guiding robot model construction with prior features*, in 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, Oct 2021, pp. 7112–7118.

Conference publications indexed by Scopus:

- 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.
  Citations: 12 (Google Scholar) / 10 (Scopus)

OTHER PUBLICATIONS

The publications listed in this section do not form the core of the thesis, but most of them are also, at least marginally, related to the thesis topic.

PUBLICATIONS IN IMPACTED JOURNALS

- J. KULHÁNEK, E. DERNER, AND R. BABUŠKA, *Visual navigation in real-world indoor environments using end-to-end deep reinforcement learning*, IEEE Robotics and Automation Letters, 6 (2021), pp. 4345–4352.
  Citations: 4 (Google Scholar) / 1 (Scopus)

- C. GOMEZ, A. C. HERNANDEZ, E. DERNER, R. BARBER, AND R. BABUŠKA, *Object-based pose graph for dynamic indoor environments*, IEEE Robotics and Automation Letters, 5 (2020), pp. 5401–5408.
  Citations: 4 (Google Scholar) / 2 (Scopus)

Conference Publications

Conference publications indexed by Scopus and by the Web of Science:

- A. C. Hernandez, E. Derner, C. Gomez, R. Barber, and R. Babuška, *Efficient object search through probability-based viewpoint selection*, in 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Oct 2020, pp. 6172–6179.
  Citations: 1 (Google Scholar)
- J. Kulhánek, E. Derner, T. de Bruin, and R. Babuška, *Vision-based navigation using deep reinforcement learning*, in 2019 European Conference on Mobile Robots (ECMR), Sep 2019, pp. 1–8.
  Citations: 36 (Google Scholar) / 17 (Scopus)
- A. C. Hernandez, C. Gomez, E. Derner, and R. Barber, *Indoor scene recognition based on weighted voting schemes*, in 2019 European Conference on Mobile Robots (ECMR), Sep 2019, pp. 1–6.
  Citations: 2 (Google Scholar) / 1 (Scopus)
- C. Gomez, A. C. Hernandez, E. Derner, and R. Barber, *Semantic localization through propagation of scene information in a hierarchical model*, in 2019 European Conference on Mobile Robots (ECMR), Sep 2019, pp. 1–6.