Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Control Engineering

Pattern Discovery, Learning and Detection in Time Series

Doctoral Thesis

Ing. Martin Ron

Ph.D. programme: Electrical Engineering and Information Technology
Branch of study: Control Engineering and Robotics
Supervisor: Ing. Pavel Burget, Ph.D.

Prague, June 2022

Thesis Supervisor:

Ing. Pavel Burget, Ph.D.
Czech Institute of Informatics, Robotics and Cybernetics
Czech Technical University in Prague
Technická 2
160 00 Prague 6
Czech Republic

Declaration

I hereby declare I have written this doctoral thesis independently and quoted all the sources of information used in accordance with methodological instructions on ethical principles for writing an academic thesis. Moreover, I state that this thesis has neither been submitted nor accepted for any other degree.

In Prague, June 2022

....
Ing. Martin Ron

Abstract

Machine learning tasks typically require large amount of data for training. This dissertation focuses on time series analysis, which is a frequent type of data collected in industry. The desired application is modeling and analysis of machines behavior. The behavior models require well-structured training data sets which are usually prepared manually. That is an expensive and exhausting task prone to errors. This complication limits the growth of applications of behavior models in industry, which motivated us to investigate the entire process of stochastic modeling of machines behavior to automate the deployment process as much as possible.

Our research was initiated by a task of modeling industrial robotic-manipulator behavior based on its power consumption, where we needed to segment a power-consumption time series by particular robotic operations. This analysis served as a support for research of optimal scheduling of robotic operations to reduce power consumption of robots. We expanded our target domain from the robotic power consumption to a general repetitive behavior observed in time series. Our findings are verified on the robotic use cases, but we keep our methods general enough to be applicable on wide range of industrial tasks, which was verified by successfully applying the methods on batch-heating ovens.

To reduce the manual interventions in the modeling process, we formulate the segmentation task as the motif discovery task. Motif discovery searches for repetitive behavior in time series and it has quadratic time complexity unless an approximation methods are used. One of our main contributions is a novel pre-filtering method that significantly reduces the time complexity of motif discovery to be linear for most real-life data sets, and ensures the high quality of the motifs. On the other end of the process, we propose a method of parameters-continuity preference which significantly reduces the size of the training data sets required for time-varying hidden Gauss-Markov models. These models provide great classification and inference performance, but there is rarely enough data to train them. Our contribution overcomes this issue by learning more information from less data. The entire work results in the Motif Discovery and Detection Framework which solves the whole task of machine behavior modeling.

The dissertation is structured according to the research progress. Each chapter contains the related work concerning the topic of the chapter, separate definition of the goals and discusses the findings from simulations and experiments. This elevates the modularity of our research, and the modularity of the Motif Discovery and Detection Framework, which promotes its adoption by practitioners.

Keywords: time series, Gauss-Markov process, Hidden Markov model, motif discovery, EM algorithm, few-shot learning, robot dynamics, Bayesian inference, birthday paradox, machine learning, data mining, segmentation

Abstrakt

Úlohy strojového učení typicky vyžadují velký objem dat pro učení svých modelů. Tato dizertace se zabývá analýzou časových řad, které jsou častým typem dat zaznamenávaných v průmyslu. Cílovou aplikací je modelování a analýza chování strojů. Tyto modely chování potřebují přesně strukturované trénovací datové sady, které jsou typicky připravované ručně. To je nákladný a únavný proces náchylný k chybám. Tato komplikace znesnadňuje rozšíření aplikací takových modelů chování v průmyslu, což nás motivovalo ke zkoumání celkového procesu stochastického modelování chování strojů s cílem co nejvíce automatizovat krok nasazení těchto modelů v praxi.

Náš výzkum začal úlohou modelování chování průmyslového robotického manipulátoru za použití dat o průběhu jeho spotřeby elektrické energie, kde bylo zapotřebí segmentovat časové řady spotřeby robotu na jednotlivé robotické operace. Tato analýza posloužila jako základ pro výzkum optimálního plánování robotických operací za účelem snížení spotřeby elektrické energie. Cílovou doménu našeho řesení jsme rozšířili ze spotřeb průmyslových robotů na obecné opakující se vzory chování v časových řadách. Naše výsledky jsou primárně ověřovány na případech průmyslových robotů, ale naše metody jsou obecné a jsou aplikovatelné na širokou škálu průmyslových prolémů, což jsme také ověřili na průmyslové aplikaci na vsádkových pecích.

Abychom zredukovali nutnost ručních zásahů do modelovacího procesu, zformulovali jsme zadání segmentace časové řady jako úlohu odhalování motivů (dále jen *motif discovery*). Motif discovery hledá v časových řadách opakující se vzory, a pokud není použita jeho aproximační varianta, má motif discovery kvadratickou výpočetní složitost. Jedním z našich hlavích přínosů je nová stochastická předfiltrační metoda, která významně snižuje výpočetní složitost motif discovery úlohy, a to na lineární složitost pro většinu časových řad z praxe, přičemž vysoká kvalita nalezených motivů zůstává zachována. Na druhém konci procesu detekce vzorů jsme navrhli metodu preferování spojitosti v parametrech, která značně snižuje množství potřebných trénovacích dat potřebných pro naučení časově proměnlivých skrytých Gauss-Markovských modelů. Tyto modely se vyznačují výbornými klasifikačními a odvozovacími schopnostmi, ale málokdy je k dispozici dostatek dat k jejich dobrému natrénování. Náš přínos překonává toto omezení efektivním získáváním více informace z menšího množství dat. Celá tato dizertační práce vyústila v nástroj *Motif Discovery and Detection Framework*, který řeší celý proces tvorby a aplikace modelů chování strojů.

Dizertace je strukturovaná podle postupu výzkumu. Každá kapitola obsahuje rešerši současné problematiky týkající se tématu dané kapitoly, samostatnou definici cílů a také diskuzi o výsledcích simulací a experimentů. Tato struktura podtrhuje modularitu našeho výzkumu stejně jako i modularitu nástroje *Motif Discovery and Detection Framework*, což má za cíl podpořit přejímání prezentovaných výsledků do praxe.

Klíčová slova: časové řady, Gauss-Markovský proces, skrytý Markovský model, motif discovery, EM algoritmus, few-shot učení, dynamický model robotu, Bayesovská inference, narozeninový paradox, strojové učení, dolování dat, segmentace

Acknowledgements

I want to thank the academia for keeping the dream of scientific thinking still alive – I do not take it for granted.

My thanks go to my family, who supported me in good faith that I would achieve my goals. Every tiny thing you did instead of me was a small step you allowed me to take in my PhD studies. I will remember that.

I thank my supervisor Pavel Burget, who was there for me when publishing new ideas and guided me in the environment of academia when I was still a scientific child. I will pass-on your support.

Thanks go to my colleague Petr Cezner, who implemented many of my (often counter-intuitive) ideas contributing to our motif discovery algorithm. I wish you good luck in your next research.

I also acknowledge all the funding grants that enabled me to assign more of my time to the research, namely:

- Grant Agency of the Czech Technical University in Prague, grant no.:
  - SGS13/209/OHK3/3T/13
  - SGS16/233/OHK3/3T/13
  - SGS19/175/OHK3/3T/13

List of Tables

List of Figures

List of Acronyms

AUC	Area under curve. 34, 35
CHIME	Collaborative hierarchy based motif enumeration. xi, 26, 34–36
CS-HGMM	Continutous state hidden Gauss-Markov model. 5, 40, 56–58
CUSUM	Cumulative summation. 42, 43, 47
DBSCAN	Density-based spatial clustering of applications with noise – clustering algorithm. 40, 45, 46
DFT	Discrete Fourier transform. 25
DOF	Degrees of freedom. 1, 53, 58, 61, 63, 79, 80
DTW	Dynamic time warping. 25, 40, 79
DWT	Discrete wavelet transform. 25
EM	Expectation-maximization. 5–7, 40, 61, 64–67, 69, 70, 77, 80, 81
GMM	Gaussian mixture model. 40
GP	Gaussian process. 82
HGMM	Hidden Gauss-Markov model. 61, 62, 67, 77, 82
HMM	Hidden Markov model. 1, 5, 51, 56, 61, 63, 79–81
KNN	K nearest neighbors. 51, 72
MoDiF	Motif discovery framework. 27, 37
NILM	Non-intrusive load monitoring. 39, 40, 62, 63
OPTICS	Ordering points to identify the clustering structure – clustering algorithm. 45, 46, 48, 79, 81
PAA	Piece-wise aggregate approximation. 25, 34
pdf	Probability density function. 3, 5, 67, 85
PLC	Programmable logic controller. 9
pmf	Probability mass function. 3, 53, 56

ROC Receiver operating characteristic. 34

SAX Symbolic aggregate approximation. 25, 26, 31, 34

SNR Signal to noise ratio. 14

TI-HGMM Time-invariant hidden Gauss-Markov model. 72

TSC Time series classification. 57, 81

TV-HGMM Time-varying hidden Gauss-Markov model. 65, 72–75, 77, 80–82

Contents

Chapter 1

Introduction

This dissertation started as an industrial-application project. Originally, the goal of the project was to optimize energy consumption of robotic cells. For that, an energy-consumption model of robotic manipulators was required, as shown in [1] and [2]. We performed consumption measurements of industrial manipulators in laboratory and also at a car manufacturer welding line (Škoda Auto). This data served us as a basis for the models, but it was necessary to carefully prepare them for any model-identification task that followed. At the beginning, we were preparing the data manually, which proved to be very slow, prone to human errors and the overall quality of the resulting data sets was sub-optimal. This became obvious especially when we started evaluating the model performance and predictions accuracy. The better data set was used, the better predictions were and vice versa. At that point, we began to research ways how to automate the process of data-set preparation and we realized there is still much to research. The need for automatic data preparation and the follow-up machine-learning tasks span across many domains, not only the domain of industrial robots. Virtually any domain that produces some time series (i.e., a time sequence of measurements) can benefit from modeling the time series. The first challenge to overcome is the data-set preparation. A high-quality data set opens the road to more sophisticated models with better performance.

This dissertation covers the whole process of building a predictive model of behavior discovered in time series. It is based on research focused on industrial robots, but our results are general and modular, they can be applied on a wide range of use cases. The thesis is structured so that each chapter is a piece of a puzzle ordered the same way as the final data-processing pipeline. In Chapter 2, we describe a constructive bottom-up analysis of the dynamic model of a 6-DOF robotic manipulator. This analysis provides valuable insight into the expected structure of the behavior model we wish to automatically learn in later stages of the work. Chapter 3 presents the first approach of manual segmentation of time series and discusses the challenges we encountered. In Chapter 4, the manual preparation is automated by formulating the task as the motif discovery problem. We introduce a fast motif discovery framework that utilizes a novel prefiltering algorithm for an efficient pruning of the search space. Chapter 5 studies the problem of organizing the unequal-length segments into an unknown number of clusters according to their similarity. A novel similarity measure is proposed that compares unequal-length sequences and normalizes them by their information content. At this point, all tools for automatic data set preparation are developed. Next chapters consider the dynamic model and its automated learning. Chapter 6 studies suitability of the Bayesian probabilistic modeling framework for the time series modeling. Hidden Markov models (HMM) are proposed as the model because they are very similar to the linear state-space models which are well adopted across industry. Applicability of the time-varying form of the HMM is demonstrated on laboratory experiments showing promising results. In Chapter 7, we adopt the more general model of continuous-state time-varying hidden Gauss-Markov model. We discuss their disadvantage of requiring a big training data set and propose a solution in a form of the parameters-continuity assumption. Extensive experiments on real-life data sets show that this assumption leads to a better performance of models even if the training data set is small. This chapter also presents the classification module which is used as the detection module of the overall data-processing pipeline.

1.1 Goals

We aim to build an entire system that simplifies and automates the process of modeling the behavior of cyber-physical systems to supervise the correct operation of the system in real time. The ultimate goal is to create a framework that consumes a sensory data stream on the input, discovers behavior patterns in it, learns a stochastic model of particular patterns and applies the models for the online pattern detection. All this process should be without any or almost any manual intervention. This is summarized in the following goals.

- Design methods for automatic search of apriori-unknown patterns in time series.
- Propose a solution to cluster the variable-length sequences to reduce the number of the unsorted patterns into groups containing only patterns which are similar.
- Develop machine-learning methods for the identification of the model parameters.
- Apply the invented methods for an online detection of repetitive behavior observed in time series.
- Validate the results on real-life use cases from industrial robotics.

It can be seen the dissertation focuses on a broad spectrum of problems, whose certain aspects are often neglected in their complexity. This dissertation provides a holistic view to formulate cross-functional requirements and proposes a solution to them, e.g., the pattern-learning module requires the motifs to be localized with a high precision.

1.2 Outline of Concept

Figure 1.1 outlines the elements of the target solution in a higher level of abstraction.

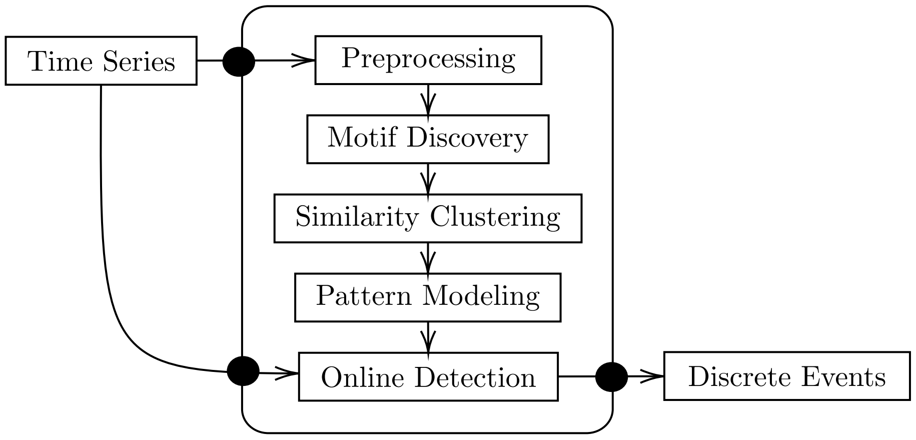


Figure 1.1: Conceptualization of framework.

Our starting point is a long data stream, i.e., time series. Mostly, we focus on uniformly sampled time series continuous in values (captured by floating point representation of numbers). The time series is analyzed, a model is learned from it, and after deployment, these models transform the time series into a much sparser log of discrete events, representing the detected patterns. The discrete-events log is our endpoint and our research does not investigate consequent analysis of these logs.

Our approach identifies several main steps in the process as depicted in Figure 1.1. Preprocessing serves as preparation step, where smoothing, standardization, whitening and other pure signal augmentations take place.

After that, we need to extract repeating patterns from the signal. That is the purpose of the Motif Discovery step. Its task is to find repeating patterns (motifs) in signal, which are not known in advance. It is a rather complicated and resource-heavy step known for its quadratic time complexity. However, a good localisation of the discovered motifs is needed because the Pattern Modeling task is very sensitive to knowing the location of the motifs precisely.

When motifs are extracted from the time series, they have to be clustered by their classes. Clustering is a classic machine learning task. According to our goals, we require the clustering to be easy to tune. Clusters can be labeled to provide better interpretability of the motifs. The output of this step is a training data set containing the clustered motifs in the time series.

Pattern Modeling takes the motifs data set and learns the model for each cluster. We prefer Bayesian models for their uncertainty specifications they provide along with their predictions and inferences.

The final step of Online Detection uses the trained set of pattern models and uses it for segmentation and following classification of the segments.

1.3 Notation

As this work intersects the domains of machine learning, dynamic system control, and signal processing, each chapter contains its specific notation that relates to the scope of the chapter. However, a subset of math notation rules applies for the whole work. Bold font of small letters is used for vectors, bold capitals for matrices, and calligraphic font for sets. Column-vector notation is used implicitly.

For denoting probability throughout this text, we use a shortened notation of, e.g., $Pr(\mathbf{X} = \mathbf{x}) = p(\mathbf{x})$, which improves readability. Probability is here thought of as a scalar function assigning a scalar value to a point from its possibly multidimensional domain space. For discrete domains, the $p(\mathbf{x})$ represents probability mass function (pmf), whereas for continuous domain, it is a probability density function (pdf).

The vertical line denotes the condition, e.g., $p(x|y)$ is the pdf of value x given value y. Where possible, we use the Kalman-filter-theory subscripts for brevity, e.g., $\boldsymbol{\mu}_{t|t-1}$ denotes an estimate of $\boldsymbol{\mu}$ at time t based on data from time instance $t-1$.

Chapter 2

Model of Robot Dynamics

2.1 Introduction

The main focus of this dissertation is on modeling robotic operations. As described in [3], it is practical to assume the robotic operations are in fact a Gaussian process with latent (hidden) variables. Many tools for analyzing the Gaussian pdf are available. A theoretical formalism for developing the hidden Markov models (HMM) and their training was described in [4]. Later, a generalization to continuous-state hidden Gauss-Markov models (CS-HGMM) was devised using a generalized form of the Viterbi algorithm [5]. In this chapter, we start by briefly introducing the physical model of the robot dynamics that is well-founded and widely used in practice. It drives our decisions throughout this dissertation. The physical model of the dynamics serves as a starting point for the expected structure of the stochastic model.

2.2 Chapter Specific Notation

The following notation is used in this chapter.

- Time derivative $\frac{d}{dt}$ is expressed by a dot above the variable, e.g., $\frac{d\mathbf{x}}{dt} = \dot{\mathbf{x}}$.

2.3 Expected Structured Model

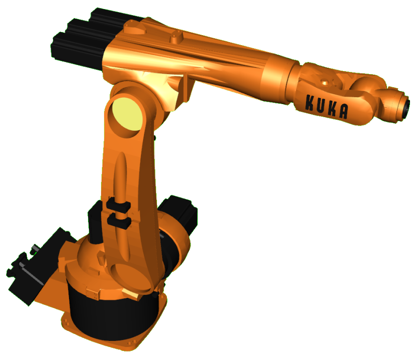


Figure 2.1: 6-DOF robotic manipulator KUKA KR5

The expectation-maximization (EM) algorithm described later in section 7.2.3 requires a structured model of the system to automatically estimate the unknown parameters of the underlying model. The system consists of dynamics of the robot and dynamics of the controlling input forces at each axis. There is a well-founded basis for modeling the dynamics of the robotic manipulator in a form of an actuated serial chain for example in [6]–[8] and specifically for the robot used in this project KUKA KR5 (Figure 2.1) was modeled in [9].

Dynamics of a serial chain manipulator can be generally described by matrix-vector form as:

$$\boldsymbol{\tau} = \mathbf{D}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{f}(\dot{\mathbf{q}}) + \mathbf{g}(\mathbf{q}) + \mathbf{f}_{ext}(\mathbf{q}), \tag{2.1}$$

where each row corresponds to one joint of the robot. Let n be the number of degrees of freedom of the robot. Variables $\boldsymbol{\tau}, \mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}$ are all vector-valued $n \times 1$ functions of time and represent joints torque, position, speed and acceleration respectively. $\mathbf{D}(\mathbf{q})$ is $n \times n$ symmetric positive definite matrix of inertia. $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})$ is $n \times n$ matrix of centrifugal and Coriolis forces. Note that both matrices $\mathbf{D}$ and $\mathbf{C}$ are in fact matrix-valued functions depending on joints configuration. Vector $\mathbf{f}(\dot{\mathbf{q}})$ represents Coulomb and viscose friction of dimension $n \times 1$, $\mathbf{g}(\mathbf{q})$ is a $n \times 1$ vector of torques induced by gravity. $\mathbf{f_{ext}}(\mathbf{q})$ is a $n \times 1$ vector of external forces applied on the joints of the robot.

Let us assume that the friction force $\mathbf{f}(\dot{\mathbf{q}})$ is negligible to simplify the model. The component $\mathbf{f}_{ext}(\mathbf{q})$ becomes a part of $\mathbf{D}(\mathbf{q})$ because only a rigid connection of manipulated object to the last link of the robot is assumed here. No other external forces are taken into account. These steps produce a simplified model

$$\boldsymbol{\tau} = \mathbf{D}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}). \tag{2.2}$$

This can be rewritten into the following form.

$$\begin{bmatrix}\dot{\mathbf{q}}\\ \ddot{\mathbf{q}}\end{bmatrix} = \begin{bmatrix}\dot{\mathbf{q}}\\ -\mathbf{D}^{-1}(\mathbf{q})\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}}\end{bmatrix} + \begin{bmatrix}0\\ \mathbf{D}^{-1}(\mathbf{q})\left[\boldsymbol{\tau} - \mathbf{g}(\mathbf{q})\right]\end{bmatrix}, \tag{2.3}$$

This form is non-linear, for the purposes of EM algorithm linear approximation is a more suitable. For linear approximation in proximity of the executed trajectory (operating point), we introduce following substitution.

$$\begin{aligned}\mathbf{x} &= \begin{bmatrix}\mathbf{x}_1\\ \mathbf{x}_2\end{bmatrix} = \begin{bmatrix}\dot{\mathbf{q}} - \dot{\mathbf{q}}_{tr}\\ \ddot{\mathbf{q}} - \ddot{\mathbf{q}}_{tr}\end{bmatrix} \in \mathbb{R}^{2n}, &(2.4)\\ \mathbf{u} &= \boldsymbol{\tau} - \boldsymbol{\tau}_{tr} - \mathbf{g}(\mathbf{q}_{tr}) \in \mathbb{R}^{n}, &(2.5)\end{aligned}$$

where $\boldsymbol{\tau}_{tr}$ is the torque of operating trajectory, $\mathbf{q}_{tr}$ is the joints value of operating trajectory and dotted versions are their respective time derivatives.

Linear approximation along a parametric trajectory is

$$\begin{bmatrix}\dot{\mathbf{x}}_1\\ \dot{\mathbf{x}}_2\end{bmatrix} = \begin{bmatrix}0 & \mathbf{I}\\ -\mathbf{D}^{-1}(\mathbf{q}_{tr})\mathbf{H}(\mathbf{q}_{tr},\dot{\mathbf{q}}_{tr},\ddot{\mathbf{q}}_{tr}) & -\mathbf{D}^{-1}(\mathbf{q}_{tr})\mathbf{C}(\mathbf{q}_{tr},\dot{\mathbf{q}}_{tr})\end{bmatrix}\begin{bmatrix}\mathbf{x}_1\\ \mathbf{x}_2\end{bmatrix} + \begin{bmatrix}0\\ \mathbf{D}^{-1}(\mathbf{q}_{tr})\end{bmatrix}\mathbf{u}, \tag{2.6}$$

where $\mathbf{I}$ is an identity matrix. The final form of the stochastic Gauss-Markov model of the overall model should be in a time-varying form for the reasons discussed above. Since a specific form of EM algorithm for the model parameters estimation was derived in this project, the model (2.2) needs to be modified to fit the suitable form of

$$\begin{aligned}\mathbf{s}_t &= \mathbf{A}_t\mathbf{s}_{t-1} + \mathbf{v}_t, &(2.7)\\ \mathbf{y}_t &= \mathbf{B}_t\mathbf{s}_t + \mathbf{e}_t, &(2.8)\end{aligned}$$

where $\mathbf{s}_t$ is the vectors of hidden states of the system. $\mathbf{A}_t$ is a state evolution matrix of the linearly approximated system at the time instance t. The vector $\mathbf{v}_t, \mathbf{e}_t$ is a process noise and a measurement noise respectively, which are both a Gaussian white noise with zero means and some parametric covariances. $\mathbf{y}_t$ is an $m \times 1$ output measurement vector and $\mathbf{B}_t$ is an output weighting matrix at time instance t. The expected models (2.7) and (2.8) are discrete in time.

Superscripts *cont* and *discr* are used to distinct between continuous and discrete version of equations respectively.

Let us form a superstate $\mathbf{s}_t$ as

$$\mathbf{s}_t = \begin{bmatrix}\mathbf{x}_1\\ \mathbf{x}_2\\ \mathbf{u}\end{bmatrix} = \begin{bmatrix}\mathbf{q}-\mathbf{q}_{tr}\\ \dot{\mathbf{q}}-\dot{\mathbf{q}}_{tr}\\ \boldsymbol{\tau}-\boldsymbol{\tau}_{tr}\end{bmatrix}. \tag{2.9}$$

The equations (2.6) can be rearranged to the form of (2.7) as follows.

$$\mathbf{A}_t^{cont} = \begin{bmatrix} 0 & \mathbf{I} & 0\\ -\mathbf{D}^{-1}(\mathbf{q}_{tr})\mathbf{H}(\mathbf{q}_{tr},\dot{\mathbf{q}}_{tr},\ddot{\mathbf{q}}_{tr}) & -\mathbf{D}^{-1}(\mathbf{q}_{tr})\mathbf{C}(\mathbf{q}_{tr},\dot{\mathbf{q}}_{tr}) & \mathbf{D}^{-1}(\mathbf{q}_{tr})\\ \mathbf{S}_q & \mathbf{S}_{\dot{q}} & \boldsymbol{\Gamma}\end{bmatrix}, \tag{2.10}$$

where matrices $\mathbf{H}(\mathbf{q}_{tr},\dot{\mathbf{q}}_{tr},\ddot{\mathbf{q}}_{tr})$, $\mathbf{C}_2(\mathbf{q}_{tr},\dot{\mathbf{q}}_{tr})$ are some (in general) nonzero parameters derived using linear part of the Taylor series of (2.3). The matrices $\mathbf{S}_q$ and $\mathbf{S}_{\dot{q}}$ represent a control law depending on system states $\mathbf{x}_1$, $\mathbf{x}_2$. $\boldsymbol{\Gamma}$ is a block matrix of a linear autonomous evolution of the control inputs.

When the system is controlled by a state feedback, matrices $\mathbf{S}_q$ and $\mathbf{S}_{\dot{q}}$ become some constants and matrix $\boldsymbol{\Gamma}$ zeroes out. Since there are some neglected dynamics of the actuating subsystems, it is likely that $\boldsymbol{\Gamma}$ cannot be zero even in the case of the state feedback. On the other hand, if the robot is controlled by a set of dynamic controllers, each looped with only one actuator, then all matrices $\mathbf{S}_q, \mathbf{S}_{\dot{q}}, \boldsymbol{\Gamma}$ can be nonzero. Although a part of dynamics would be neglected this way, it reduces complexity of the learning problem for the EM algorithm.

Discretization of $\mathbf{A}_t^{cont}$ using the forward Euler approximation method yields

$$\begin{aligned}\mathbf{A}_t^{discr} &\approx (\mathbf{I}+\mathbf{A}_t^{cont})h\\ &= \begin{bmatrix} \mathbf{I} & \mathbf{I}h & 0\\ -\mathbf{D}^{-1}(\mathbf{q}_{tr})\mathbf{H}(\mathbf{q}_{tr},\dot{\mathbf{q}}_{tr},\ddot{\mathbf{q}}_{tr})h & \mathbf{I}-\mathbf{D}^{-1}(\mathbf{q}_{tr})\mathbf{C}(\mathbf{q}_{tr},\dot{\mathbf{q}}_{tr})h & \mathbf{D}^{-1}(\mathbf{q}_{tr})h\\ \mathbf{S}_q h & \mathbf{S}_{\dot{q}} h & \mathbf{I}+\boldsymbol{\Gamma}h\end{bmatrix},\end{aligned} \tag{2.11}$$

where h is a sufficiently short sampling period. This is the expected time-varying discrete system matrix. First block row is set and known and the rest are subjected to the EM algorithm estimation.

Clearly the choice of the discrete approximation has a substantial impact on the resulting form of $\mathbf{A}_t^{discr}$, but the purpose of this expression is just to find acceptable initial conditions for subsequent iterative estimation. However, an analysis of parameters found by EM estimation could increase precision of robot identification.

The expected linearly approximated measurement matrix $\mathbf{B}_t$ is derived from a simplified mathematical model of synchronous electric motor as shown in [9] and takes a form of

$$\mathbf{B}_t^{discr} = \mathbf{B}_t^{cont} = \begin{bmatrix}0 & \boldsymbol{\Lambda}_{\dot{q}} & \boldsymbol{\Lambda}_{\tau}\end{bmatrix}, \tag{2.12}$$

where both $\boldsymbol{\Lambda}_{\dot{q}}$ and $\boldsymbol{\Lambda}_{\tau}$ are some generally nonzero parametric matrices representing an influence of a self-induced current and a control input respectively. The output measurement should not be affected by a position of the motor, but only by the speed of the motor and then by some reference signal, which is described as a Markov process here. A reasonable simplification of $\mathbf{B}_t = \mathbf{B}$ can additionally reduce the parameters learning complexity.

2.4 Conclusion

In this chapter, we described the expected structure of the robot dynamical model. The parameters $\mathbf{A}_t^{discr}$ and $\mathbf{B}_t^{discr}$ from (2.11) and (2.12) are quite complicated. Obtaining them by a bottom up modeling from the first principle is prone to error. This is the main reason why the machine learning approach for identifying the parameters is so attractive. However, machine learning is known for requiring a lot of data. In this dissertation, we focus on solving this issue by automating the training-data acquisition and by reducing the required amount of them.

Chapter 3

Manual Preparation of Data Sets

The first step towards our goals of automatic behavior detection is a semantic segmentation of time series, i.e., selecting segments that correspond to a particular behavior of a system. The traditional approach requires user to manually specify approximate templates for the segments, and then directly detect the patterns based on the similarity with the specified template. This chapter builds on our initial publication [10] and our work published in [11], where we proposed a significant speedup of the similarity search while utilizing a similarity measure that is well interpretable and thus easy to tune. The burden of manual preprocessing of time series can be solved by motif discovery which is studied later in Chapter 4.

3.1 Introduction

Manufacturers today concentrate more and more on the energy savings and one of the ways to achieve lower energy consumption is the possibility to bring devices (and especially robots) into a sleep mode. Estimations and some preliminary measurements show that bringing the robots into a sleep mode during the weekends, when the production is stopped, offers potential savings in the range of percents out of the total consumption of a production lines. Such type of pauses is called *planned*. During working time there may however occur pauses that are *unplanned* and the possibility to use the sleep mode also here offers even higher energy reductions. However, mechanisms are needed to detect whether there are pauses in the production and if it makes sense to utilize an energy-saving mode.

The usual case we focus on is following. The robotic cell is set and configured and running. An uninterrupted production is the priority and thus no or very limited external interventions are allowed. Any change of the controlling program or a robotic program would require successive testing, which would prolong pause in production and also increase cost of deployment of the solution. Complexity of the controlling PLC programs grows with the complexity of the cell and intervention of more programmers also increases the risk of introducing additional errors. This is why it is hardly possible to insert additional code into controlling PLC to enable observation of the process. On the other hand an independent modular estimator of the operating states based only on measured power consumption does not require any intervention in the current control logic. This brings valuable possibility to speed up the deployment of solution on the existing robotic line without risk of introducing new errors into original programming of devices in line.

This chapter focuses on the analysis of the robotic movements, which are called operations, in a welding cell based on the measurement of the robot power consumption. In such a way it is possible to have information about the production status of the welding cell, i.e. of the individual robots, without a single intervention or disruption of the main controller program that controls the welding cell. Methods of behaviour pattern recognition are used.

These methods of pattern recognition are limited by the initial pattern learning. Right now our focus is on subsequent procedure of localisation of patterns as if they were properly learned. The learning mechanism is substituted by manual selection of proper patterns to be searched in the online data. An automated machine learning of the patterns will be of interest in the upcoming phases of this work.

The possibility of knowing the robotic operations and their power consumption allows opening further research directions such as automatic identification of the state machines, which would describe the behaviour of the robotic cell, or such as adapting an eventual energy function obtained analytically.

3.1.1 Objectives

The welding cell this use case is focused on, consists of 6 robots and other pieces of equipment such as turn tables, conveyors, gluing machine and welding guns. A general structure of the welding cell is depicted in Figure 3.1.

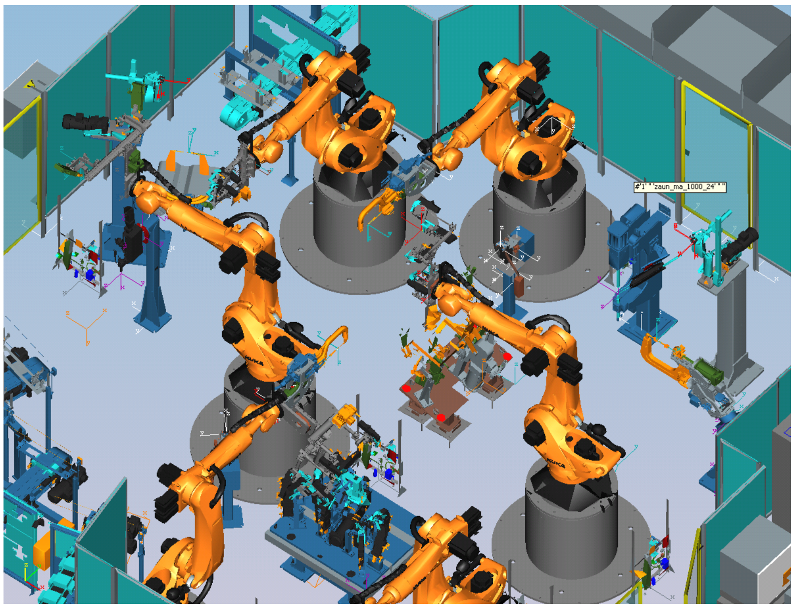


Figure 3.1: Structure of the welding cell

Figure 3.2 shows power consumption of one robot from the cell in Figure 3.1 (blue line), and the power consumption of the turn table (red line) that is at the beginning of the cell's product flow. The turn table consumption is used in further assumptions to identify the beginning of the next production cycle, i.e. the fact that a new product has been inserted and that the sequence of robot operations for this product can start.

The main objective of the chapter is to present a way how the behavior of the production line can be analyzed based on the measurements of the electrical power consumption of the individual components. Moreover, the power consumption must be segmented into individual operations to analyze them separately. This separate analysis is performed also in the simulation environment and can be utilized for example to obtain a so-called energy function, which can be used in further optimizations as described, e.g., in [2]. More details are provided in Section 3.3.

In our case, the electrical power consumption has been measured for every robot and also for other major energy consumers. This extensive measurement had to be done in order to identify the major consumers for which it makes sense to measure the electrical power, i.e., whose power consumption is worth optimizing.

Another important issue is the potential of power consumption decrease by bringing the devices into a sleep mode during their inactivity. This capability of certain devices such as

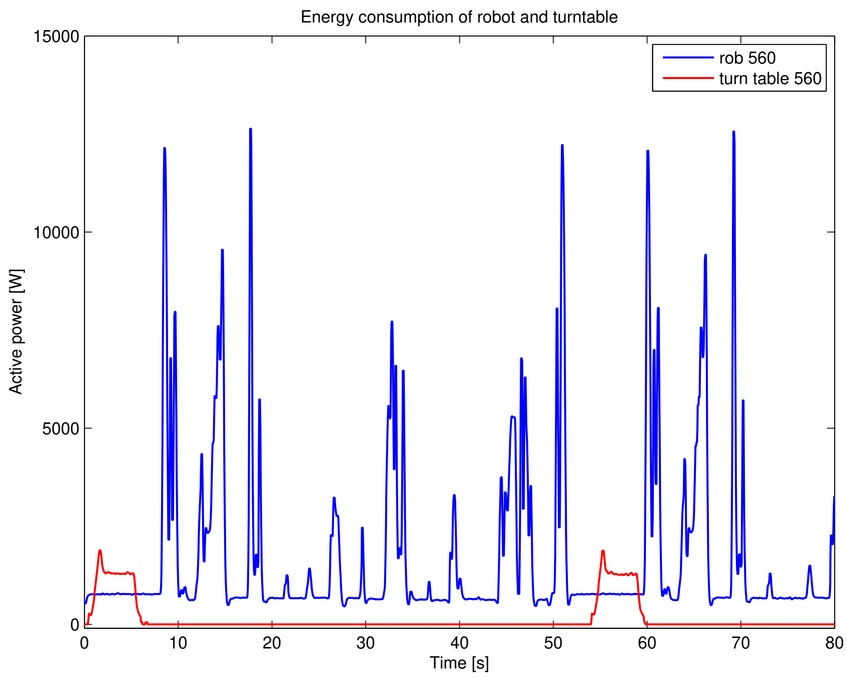


Figure 3.2: Power consumption of one robot

robots can be used during short pauses during production (starting in range of tens of seconds) to longer pauses (typically pauses during the weekends).

3.1.2 Related Work

In paper [12] the authors use region-based segmentation stemming from frequency analysis of the original signal. This brings good results as the authors try to identify anomalies in the signal that are composed of different-frequency signals and thus produce regions various from those representing the correct signal.

Paper [13] presents an approach where a finite state machine is built based on the discrete events in the signal representing the sum of the power consumption of the appliances. The consumption of the appliances or their count are unknown. The consumption of each appliance is supposed to be constant in each state. For a given measured consumption a combination of appliances and their states must be found that correspond to that consumption. In other words, a scalar value is assigned to each state of each appliance. This approach cannot be used in our case because the power consumption of the robots varies during the execution of their operations and the operations may not be characterized with scalars. Other papers like [14], [15] or [16] concentrate on home appliances as well. Paper [17] focuses on behavior pattern recognition in long-term time series data obtained from measurement of power consumption in urban zones. The authors use Pearson correlation coefficient as the similarity metrics and Ward's linkage for hierarchical cluster analysis.

An industrial use case is dealt with in paper [18] that relates to a state estimation and a corresponding energy audit of injection moulding machines. The authors concentrate on identifying the production state of the machine and neglect its sub-states. The reason for neglecting the sub-states is that their duration is too short (much shorter than the duration of the production state, which may be down to 14 seconds, according to [18]), and typically also the power level in different states differs significantly. The two-level neuron network classifies the states and is not able to classify the sub-states because of the length of the sample window that is used for classification. Authors of paper [19] focus on pattern recognition of 1-D signal in industrial batch dryer. Their goal is to slice the measured data of pressure into time windows of the periodic batch processing intervals. They use supervised learning of a Takagi-Sugeno fuzzy model. Their data are strongly periodic and interval boundaries are distinct. This way cannot be used for our case because the operational states of the robots must be identified with a much higher granularity, i.e. the sampling period for the power consumption measurement is much shorter, and therefore we are also able to identify operational states, whose duration is in range of a few seconds.

Generally, a major research area for pattern search is the area of speech recognition. There have been many research papers dealing with this topic such as [20] or [21]. In most cases a great robustness is needed because the pronunciation of different speakers varies for individual words. This holds even for a speech of one speaker according to his/her current conditions. Therefore, in the time domain the signal is variable and the corresponding frequency spectrum of the same word may differ significantly. During the subsequent classification in the frequency domain a certain robustness is necessary to recognize such a word. Moreover, the robustness does not worsen the precision because different words are well differentiated also in the frequency domain from each other. In our case, frequency analysis is not suitable because the signals are very similar in spectrum. Figure 3.3 contains frequency spectra of two instances of the same operation

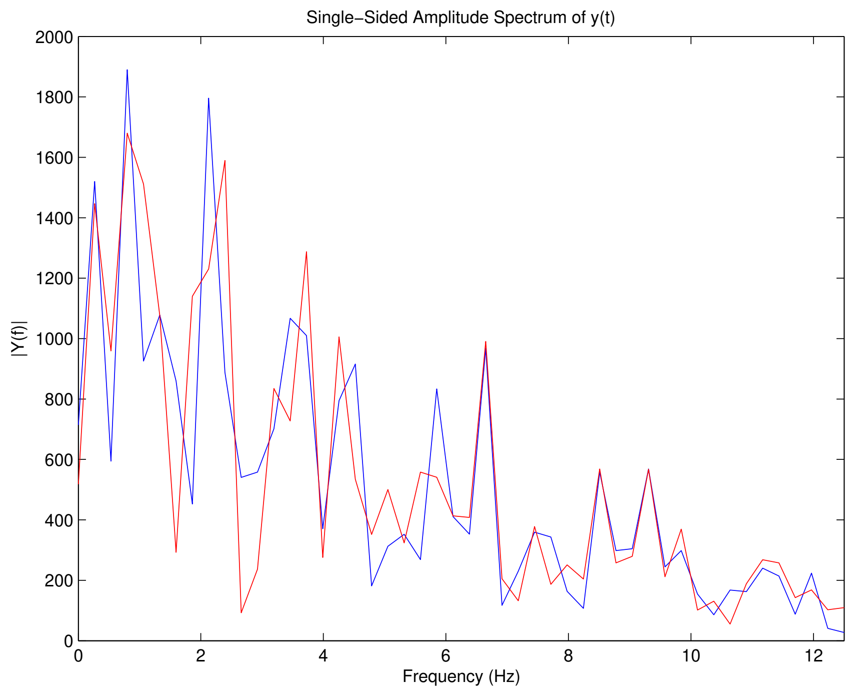


Figure 3.3: Frequency spectrum of two instances of one operation

but from different time points during the day. One time point is at the end of the working shift and the other is at the beginning of the next one. The correlation of the two instances from Figure 3.3 is 83 %, which is too low contrary to the fact that both spectra correspond to the same operation. Moreover when we analysed frequency spectra of two different operations, we got results with correlation up to 85 %. This fact shows that the frequency analysis cannot be used to classify the operations.

3.2 Identification of Operations

As described above, we have devised methods to identify the energy consumption of each separate robotic operation based only on the power measurement. By watching the welding process separable operations were observed and consequently we were able to label sections in the power data, which correspond to particular operations of each robot. One pattern for each robotic operation was chosen as a model pattern. Such a pattern is to be automatically found in whole set of energy data. This preparation steps must be done before automatic localization of patterns can proceed.

For the sake of clarity we reduce our view in the following text only on one pattern of one operation of a robot in offline data. By dividing the primary task into subproblems we achieve simplification of the procedure. First the data needs to be filtered so that we suppress the noise and the error coming from misplaced samples. Then we need to evaluate similarity of all segments of energy data and afterwards the measure of similarity is taken as a threshold to find local maxima. Location of these maxima matches the location of the pattern in the data.

Our data are rather densely sampled and when the model pattern is extracted, it leads to a long vector of many samples. This vector is said to be the feature vector. The dimension of this vector has slowing impact on computation time. We propose a reduction of the dimension of the feature vector to speed up the searching algorithm.

3.2.1 Filtering Data

As mentioned above, our data suffer from noise and random sample-swap corruption. Sample swapping cannot be completely prevented by means of sorting samples according to their timestamps. And yet this random error in data has a big impact on success of the searching algorithm. For that reason we have to bring filtering to our solution to minimize occurrence of this error. In addition we also need to suppress noise in signal by filtering.

Due to the nature of error we want to get rid of, a median filter is used. Its filtering window length was chosen empirically to filter out random signal errors with a big amplitude difference such as unsynchronised neighbouring samples, where two neighbours are swapped. Median filter is still conserving high peaks that are used as classification feature in detecting part of our work. For signal x_t, for $t = 1, 2, \cdots, T - w + 1$ with given width of window w the median filter transforms signal as follows:

$$\tilde{x}_t = Median\{x_t, x_{t+1}, \cdots, x_{t+w}\},$$

where $\tilde{x}_t$ means median-filtered sample on t-th position, and T is the total number of samples in the input signal. The filtered signal $\tilde{\mathbf{x}}$ is defined as a column vector:

$$\tilde{\mathbf{x}} = [\tilde{x}_1, \cdots, \tilde{x}_t, \cdots, \tilde{x}_{T-w+1}]^\top. \tag{3.1}$$

As we need the detection algorithm to be fast enough to satisfy real-time demands, we must choose the filtering window to be short. A window of length $w = 5$ is chosen as a good balance between smoothing effect and computation cost. The longest usable window was $w = 7$ samples, above this number of samples signal distortion collides with other threshold rules introduced in our algorithm.

3.2.2 Similarity of Segments

We acquired labelled model patterns by manually choosing well measured filtered segments of energy signal. Next task is to find a good measure of similarity when we are given two vectors of the same length containing 1-D data signals. As discussed in the introductory section, a frequency analysis does not result in very convincing outputs in our case. Therefore the similarity of the signal in time domain is evaluated in a similar way as the image-processing techniques do.

From patterns located manually from the energy data we expect variations between them to be caused by static gain and additive white noise. The observed static gain is most likely correlated with a continual use of robots – all the mechanisms in the production cell reach higher temperature and their power consumption rises. The trend of consumption increase is not linear, but as it is very slow, we model it as if it was simply linear. The two column vectors $\mathbf{x}, \mathbf{y}$ of the length w of the same pattern shape are expected to satisfy equation:

$$\mathbf{y} = a\mathbf{x} + b + \mathbf{e}_t, \tag{3.2}$$

where a is a constant, b is a constant and $\mathbf{e}_t$ is a column vector of normal random variables representing the white noise, being dependent on time. For signals, which are just affine projection of each other, their correlation coefficient is a good measure of dependency. In fact the Pearson's correlation coefficient is equal to one for such a dependency. Our expected form of dependency includes the white noise element in equation (3.2). This noise will lower the correlation coefficient as the SNR (Signal to Noise Ratio) goes to zero. But if we expect rather small values of noise (SNR much bigger than 0 dB), the correlation coefficient will not be affected much.

From this preliminary analysis we can justify use of the Pearson correlation coefficient for its simplicity and expected functionality in our case. It is defined as follows.

$$\rho_{\mathbf{x},\mathbf{y}} = \frac{cov(\mathbf{x},\mathbf{y})}{\sigma_{\mathbf{x}}\sigma_{\mathbf{y}}} \tag{3.3}$$

$$cov(\mathbf{x},\mathbf{y}) = \frac{1}{w}\sum_{i=1}^{w}(x_i - \mu_{\mathbf{x}})(y_i - \mu_{\mathbf{y}}) \tag{3.4}$$

$$\sigma_{\mathbf{x}} = \sqrt{\frac{(\mathbf{x} - \mu_{\mathbf{x}})^\top(\mathbf{x} - \mu_{\mathbf{x}})}{w}} \tag{3.5}$$

Coefficients $\sigma_{\mathbf{y}}$ and $\sigma_{\mathbf{y}}$ are the standard deviations of values of vectors $\mathbf{x}$ and $\mathbf{y}$ respectively. The $cov(\mathbf{x},\mathbf{y})$ is the cross-covariance of the vectors $\mathbf{x}$ and $\mathbf{y}$. The $\mu_{\mathbf{x}}$ and $\mu_{\mathbf{y}}$ are the mean values of $\mathbf{x}, \mathbf{y}$ respectively.

The mean values $\mu_{\mathbf{x}}, \mu_{\mathbf{y}}$ are not known exactly and we use their approximations by sample means $\bar{x}, \bar{y}$ respectively. The computation of covariance is then also modified to receive the best unbiased estimate. The formulas for sample correlation coefficient are as follows.

$$\hat{\rho}_{\mathbf{x},\mathbf{y}} = \frac{\widehat{cov}(\mathbf{x},\mathbf{y})}{s_{\mathbf{x}}s_{\mathbf{y}}} \tag{3.6}$$

$$\widehat{cov}(\mathbf{x},\mathbf{y}) = \frac{1}{w-1}\sum_{i=1}^{w}(x_i - \bar{x})(y_i - \bar{y}) \tag{3.7}$$

$$s_{\mathbf{x}} = \sqrt{\frac{(\mathbf{x} - \bar{x})^\top(\mathbf{x} - \bar{x})}{w-1}} \tag{3.8}$$

Since exact correlation coefficients are unreachable for us, we will further in this chapter use these terms always in the meaning of sample correlation, sample mean and sample deviation and we will omit the word *sample*.

3.2.3 Segment for Comparison

After having established the measure of good match of two segments for the energy data it must be considered, which segments to compare. The straightforward procedure is to compare every possible window of the same length as the model pattern vector. Such a strategy guarantees us a good precision of localization in time. It is also the most demanding way regarding the computational cost. This is where we proposed two distinct ways how to form the vectors to compare.

Straightforward Choice of Segments

In this case we proceed simply as described above. We pick the range of samples from energy data so that we have the segment of the same length as the length of model pattern. Then we calculate the Pearson correlation coefficient for them and store it in the vector of results $\mathbf{r}_{straight}$. The length of $\mathbf{r}_{straight}$ is equal to the number of repeated calculations of correlation coefficient and can be expressed as

$$\dim(\mathbf{r}_{straight}) = \dim(\tilde{\mathbf{x}}) - w + 1$$

where $\tilde{\mathbf{x}}$ is the filtered vector of energy consumption data defined in equation (3.1). The bigger w is, the less correlations are needed to be computed, but also the harder it is to compute them. Number of multiplications for covariance computation is w and also number of multiplications for the deviations is w for each. Roughly $3\,w$ multiplications need to be done for each checked window on the energy data vector. This leads to approximately $3\,w\dim(\mathbf{r}_{straight})$ multiplicative calculations on floating point numbers to search through the whole data vector $\tilde{\mathbf{x}}$. For our data holds that $w \ll \dim(\tilde{\mathbf{x}})$ and thus we can simplify that the computational cost of our algorithm for correlation evaluation depends approximately linearly $\mathcal{O}(\dim(\tilde{\mathbf{x}}))$. The length of $\tilde{\mathbf{x}}$ cannot be changed, since we still need to search through all the data.

Feature Extraction

As we showed in subsection 3.2.3, reducing the length of the model pattern algorithm would lower the computational cost of computing correlation vector $\mathbf{r}_{straight}$. With increase of speed we would loose portion of certainty this way, because we would give up portion of information contained in the trimmed part of the model pattern. However, the demand for the speed of algorithm is unavoidable. It may be bearable to let the computations run for some hours on static data for one robotic cell, but when confronted with higher number of cells running online, the speed becomes critical.

For that we need to extract as much of the viable information from the original model pattern as possible. Same extraction must be done on the energy data to get the data vector of the same type of features. On the other hand the computational cost of extraction must be small, so that we improve overall performance rather than generate even more expensive one. Our energy data contain reasonable amount of local maxima and even human sight can recognise periodic repetition of these peaks. That is why we empirically decided to choose energy consumption peaks for our distinctive feature. Peaks can be detected during just one iteration per sample and so it is viable even for online execution.

To avoid undesirable peaks we bring the following additional rules.

- Only peaks with value above the threshold are considered. The threshold is chosen so that only dominative peaks are checked. Those contains enough information for classification and avoiding small peaks helps to shrink the length of feature vector.
- When more than one peak is present in defined neighbourhood, then only the biggest one is picked. First peak is prioritized for the same-valued peaks.

Due to use of the median filter the local extrema of energy data are usually flattened. The neighbourhood is empirically chosen as double of the length of the median filter window.

Fig. 3.4 shows the model pattern and peaks chosen according the rules above. The distinctive peaks under the green dashed threshold line worth special attention. Suppose we have chosen the threshold right on top of some peak as we can see in Fig. 3.4 at time $8.5\,s$. Energy data shows a small drift in offset during the working shift of the robotic cell. This drift can bring the peaks neglected before to half plane above the threshold. This would cause that some patterns physically generated by the same type of operation would have few more dominant peaks than others. This problem leads us to the next step of the boosted algorithm.

The measure of similarity we chose has its drawbacks, more precisely we need to have a pair of the vector of the same length to be able to compare them. By extraction of feature vector we actually create new downsampled data vector with adaptive sampling time. If we focus only on the case when two patterns should be classified to belong to the same operation, we can

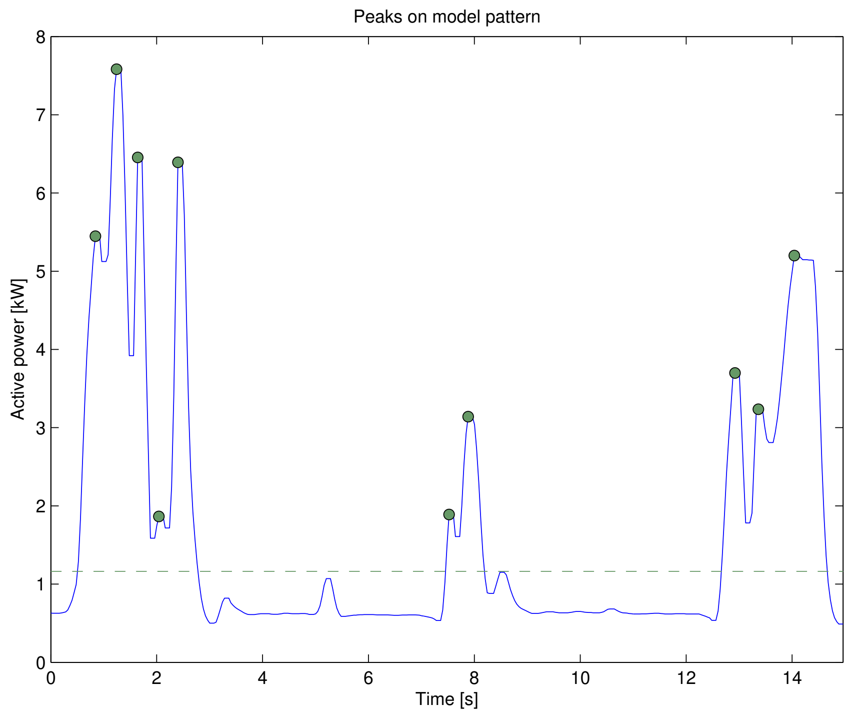


Figure 3.4: Extracted peaks from model pattern

have four distinctive cases of timestamp matching. See Fig. 3.5, where some general simplified feature vector is presented. Values are here omitted, as the focus is only on timestamps of the samples. The rows represent possible feature vectors of the pattern of the same operation, all of them are aligned by the first sample as the window check them. The top row of index 5 is our model pattern feature vector. The rest of rows (blue) are taken from energy data. We could pick simply first N samples in the window and check their correlation with model. This would work if the extraction marked only valid peaks and did not add any peaks on the border as discussed earlier. That case should occur most of the times but it depends heavily on the good choice of threshold for extraction. This simplest case is depicted in Fig. 3.5 on the row indexed by 4. The case when there sometimes are peaks added from region near the threshold is depicted by the row of index 3. Similarly row 2 shows case when some peak falls below the threshold and is missing in data (around time 3). Finally on row 1 there is mixture of both cases - missing peak and added peak.

Simple approach with considering first N samples leads to drop of magnitude of correlation coefficient on location where it should be maximal. To fix that we need to pair up corresponding timestamps together and drop those which remains unpaired. We know apriori the duration of the model pattern and the patterns we search for should not deviate much from it. So we use the window at least as long as is the model pattern by means of time duration. Maximal length is not explicitly bounded. Then we assign to each sample of model pattern vector sample from data vector based on the eucleidian distance of timestamps. In other words, we choose from data the closest neighbours to the aligned model pattern samples. This way we equalize sizes of compared vectors. Finally we evaluate correlation of the vectors of the aligned timestamps and only vectors correlated enough are passed next for value correlation. This way we consider correlation not only of order of samples but also of the position of chosen samples. This also decreases the number of correlations of value vectors to check.

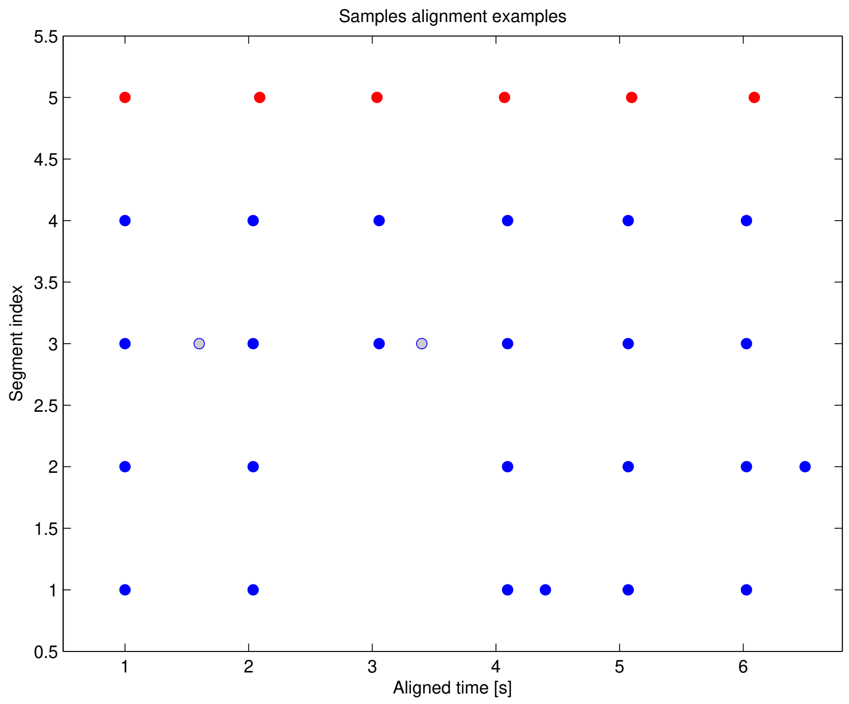


Figure 3.5: Segmentation match cases

3.2.4 Correlation Vector

The procedure above generates vector of correlation coefficients as shown on Fig. 3.6. In this vector we are looking for the values greater than some threshold. This threshold was was chosen as very sharp match, its value is 0.99 and is depicted by green dashed line in Fig. 3.6. You can see that there are two maximals, where correlation exceeds threshold value and these extrema are then classified as match.

In the Fig. 3.7 is the example how the boundaries of operations are located. Red vertical lines depicts where operation starts or ends.

3.2.5 Locating the Pauses

For overall evaluation of energy consumption for some longer period of time (days), we need to locate also pauses and determine their durations. This is much easier task than locating patterns because pauses are typically sections in energy data, where the consumption is rather constant. We compute the difference of neighbouring pairs of samples and consider only values smaller than some minimal value. Then we apply rule that constant sections shorter than 2 minutes are not considered to be actually pauses.

3.2.6 Evaluation of Hit-miss Ratio

We need to define the measure of quality of our algorithm. This is not a trivial task since our knowledge about the robotic cell is very limited (intentionally). The few things we can assume are as follows. We know the number of cycles over some period, because it is equal to the number of turns of the turntable which consumption we measured. These turns are well distinguishable so the number of turns is reliable. But during the shift the operation types varies and robot is operating based on one robotic program for a portion of cycles and then it switches for different robotic program for next portion of cycles and so on. Also some special programs are sometimes triggered, e.g., periodic tool maintenance.

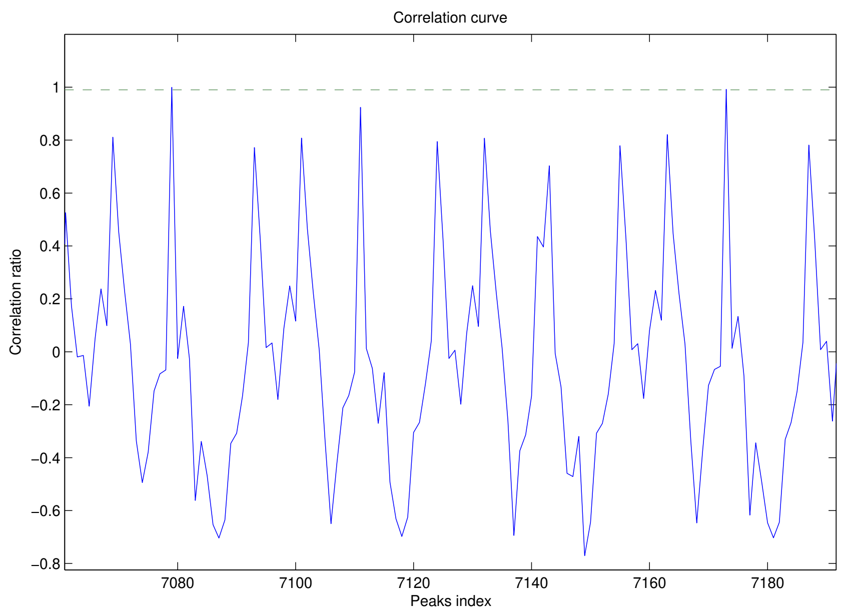


Figure 3.6: Correlation vector of peak values

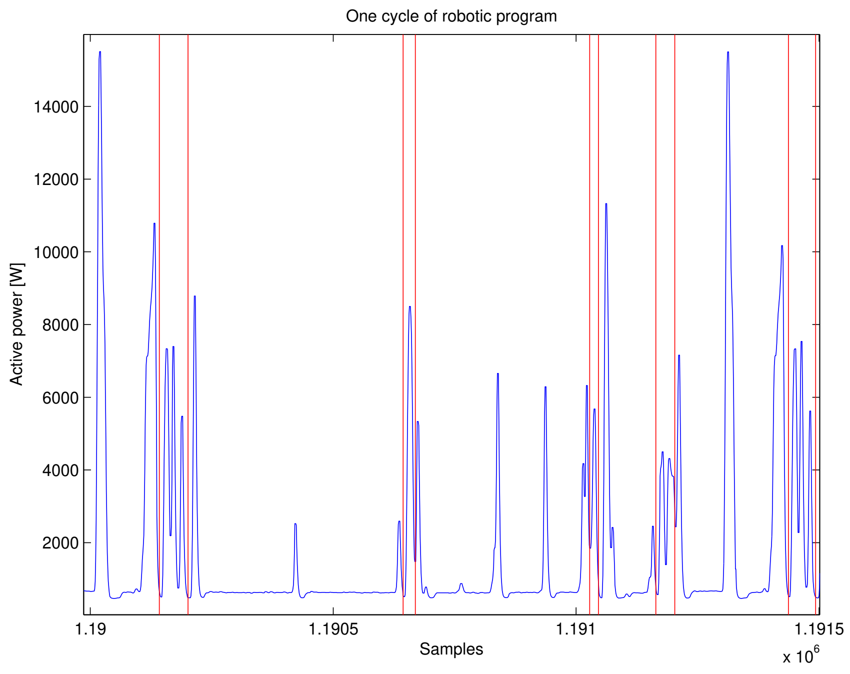


Figure 3.7: Identified operations' boundaries

We can evaluate the recall of the algorithm. Let us have n_{cycles} the number of cycles, $n_{matches}$ the number of matches for one particular operation as the operation occurs once per cycle at maximum. We define the recall – the measure of quality $q \geq 0$ of our algorithm as the ratio:

$$q = \frac{1}{n_{cycles}} \sum_{operations} n_{matches} \tag{3.9}$$

We rely here on the fact that we can exactly determine n_{cycles} from some appliance behaviour in robotic cell, but this assumption may not always hold. In these cases we use $\hat{n}_{cycles}$ estimate from tact period of the robotic cell and we rely on that we correctly classified pauses. then we can use quality measure estimate $\hat{q}$ as follows.

$$q \geq \hat{q} = \frac{1}{\hat{n}_{cycles}} \sum_{operations} n_{matches} \tag{3.10}$$

$$\hat{n}_{cycles} = \frac{T_{considered} - T_{pauses}}{T_{tact}} \tag{3.11}$$

Where we suppose that the tact duration T_{tact} is at maximum the real duration of one cycle or less. Then holds that $\hat{n}_{cycles} \geq n_{cycles}$. Parameter $T_{considered}$ is the duration of searched section of energy data and T_{pauses} is the total duration of all pauses in searched section of data. The closer q gets to the value 1, the better our locating algorithm performed. Values above 1 indicates false positive classifications, lower values indicates false negative classifications.

We tested our algorithm on all robotic operations for one program from all six robots from testing robotic cell. The length of searched data was five days of production. Our results in terms of the quality measure q reach 88 %, but when a bad model pattern is chosen, q can drop even down to 55 %. Overall, when a suitable pattern is chosen, the resulting quality measure is in range 70 % to 90 % This shows that reliability of our detection depends strongly on choice of a good representative model pattern as we discussed in subsection 3.2.3 about feature extraction.

We address these caveats of the direct similarity measure approach in later chapters (e.g., Chapter 6), where we develop a model-based approach for detecting operations.

3.3 Power Consumption Simulation

In some situations it is not possible to perform all necessary experiments and measurements with real equipment because of some physical or organizational limitations. This has also been the case of this application because all the measurements were done during ordinary production. Thus, it has not been possible to change the robotic paths to perform any additional movements, which are not part of productive robotic operations, to identify a set of parameters relating e.g. to the dependency of the power consumption on a specific robot trajectory, to the robots' dynamic parameters, etc.

Therefore a simulation environment has been used, namely Tecnomatix Process Simulate. This environment contains robot controllers implemented according to the Robot Controller Simulation (RCS) specification, which means that the robot controllers perform Realistic Robot Simulation (RRS). One of the features of the recent Process Simulate version is the possibility to simulate power consumption of the robot movements. By summation of the total energy used by a robot for one particular movement performed repeatedly with different speed settings we can construct so called energy function.

The energy function $f_E(x_i; \tau_i)$ is the primary motivation to perform the experiments, where subscript letter E stands for energy. Note that the energy function depends on the trajectory the robot moves along, which is explicitly expressed by the trajectory index τ_i. The energy function can be approximated as

$$f_E(x_i; \tau_i) = a_{-1}x_i^{-1} + a_0 + a_1 x_i \tag{3.12}$$

as suggested in [2]. Variable x_i represents the duration of the execution of trajectory τ_i. If the energy model of the robot is not known, which is the most usual case, the energy function must be estimated from measurements or from simulation. As mentioned above, the measurement with physical robots is usually not possible and therefore the robotic paths must be imported in the simulation environments and the energy function must be approximated from the simulated values.

The generation of the energy function is done in the following way. For a given robotic cycle the operations are grouped into movement and welding ones. The movement operations are those during which the robot moves from one point to another over a trajectory without any interruptions. The welding operation is composed of movements as well but the trajectories are typically short and the major activity of the robot is to perform welding at the specific welding points. The reason for such a division is the assumption that during welding operations the speed of the movements cannot be changed significantly because of the short distance between the individual welding points. On the other hand, the speed during the movement operations makes sense to be changed to minimize the power consumption. More details about this optimization task is provided in [2].

Figure 3.8 shows the Gantt diagram of a robotic operation sequence performed in the welding cell. The first horizontal line is the length of one cycle, i.e., the time after which the robot repeats its set of operations for the next part. The second horizontal line means movement of the robot to the turn table and the third line represents the welding operations. The subsequent lines represent operations of gripping the part and moving it to another table.

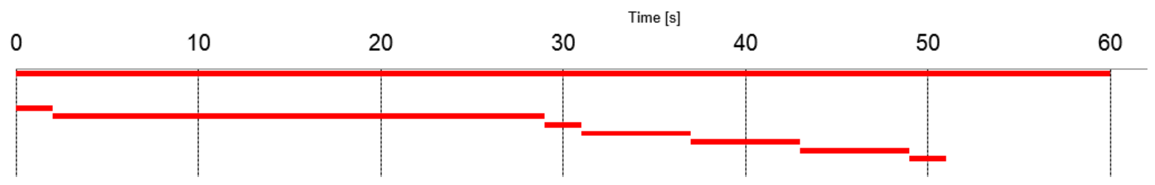


Figure 3.8: Gantt diagram example of operations of one robot in the cell

This robotic cycle can be separated into two movement and one welding operation. The welding operation is represented by third horizontal line (as described above). The first movement operation is shown by the second horizontal line, i.e., the one preceding the welding, and the second movement operation is represented by the set of the horizontal lines that succeed the welding operation.

The data obtained from the simulation are approximated with function from equation 3.12 as shown in Figure 3.9. Unfortunately the precision of the simulation can hardly be verified with measurement because of the fact that not all the trajectories, which were used to obtain the energy function in simulation, can be executed in the production cell. However, according to simulation software benchmarks, the simulated cumulative energy consumption is within 20 % margin of error, which justifies the use of the simulation results.

3.4 Conclusion

In this chapter use case, we have shown how the power-consumption signal can be used to recognize behavior patterns corresponding to individual robotic operations. This is necessary for detecting the production pauses that may be either planned or unplanned. Especially in the case of unplanned pauses, the model of the welding cell must be known to predict further behavior. The methods described here laid a basis for optimizing the robotic movements in [2] to decrease their energy consumption, by putting the robots to a sleep mode when a production pause occurs. The detection of robotic operations achieved the hit ratio of almost 90 % when tuned properly.

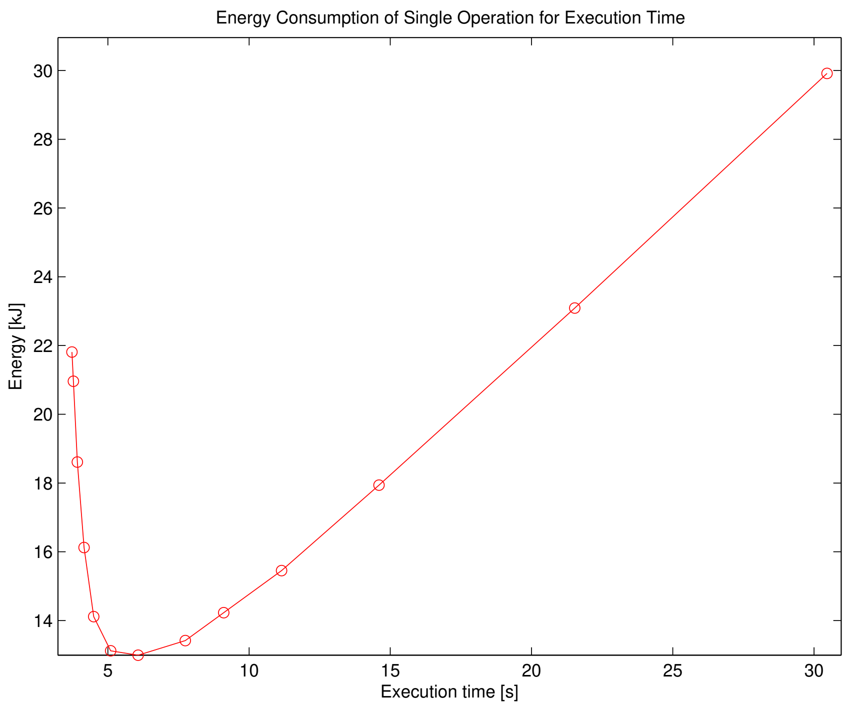


Figure 3.9: Energy function

However, detection based on direct similarity measure has its limits. Namely, it requires manual tuning of thresholds both for detection and for feature extraction. The approach is sensitive to the correct choice of the representative segment and similarity deviations are difficult to interpret.

These limitations led us to further investigate the topic of automatic segmentation, pattern modeling and detection in next chapters.

Chapter 4

Motif Discovery

Most industrial processes generate time-series data containing repetitive patterns. E.g., a welding robot repeats the same welding operations on each car body it processes. The robot's movements are measured in the form of a time series, and the welding operations make the patterns. In reality, the operations vary due to noise or drift in hidden process parameters, but their imprints in sensory time series remain recognizable. We exploit the repetitiveness of manufacturing processes and formulate the Motif discovery task. The priority is to devise an algorithm for a lossless high-precision motif localization. However, the presented algorithm is general and can be applied to the state-of-the-art Motif discovery algorithms. This chapter details our original idea of acceleration of the Motif discovery and is based on our submitted work [22] (unpublished as of yet).

4.1 Introduction

With the digitization of the manufacturing processes, a large amount of time-series data is produced. These data usually come from sensors used for the monitoring and control of manufacturing processes. The time series data are also available in many other domains such as financial markets, wearable electronics, healthcare, etc.

One of the problems in the time series analysis is finding repeated patterns, so called motifs. This process is called pattern mining or motif discovery and interest in it has risen significantly in the last two decades [23]. The term motif came from the bioinformatics domain, where it meant a repeated nucleotide or an amino-acid sequence pattern [24]. The patterns found can be used for various problems such as anomaly detection, discrete log generation, classification, or regression.

The need for motif discovery in time series has been discussed extensively. A comprehensive overview is available, e.g., in [25]. We consider motif discovery to be important for an automatic dataset preparation, which is an essential task in every machine-learning application. The dataset preparation from continuous time series is a manual-intensive task where there is no simple rule for automated segmentation and annotation. Depending on the pattern and its characteristics, various applications arise, e.g., the health of the monitored machine, anomaly detection in the financial market behavior and others.

The majority of current research into motif discovery expects the pattern, or at least its length, to be known in advance. This is a significant limitation in cases where there is a need to search for *any* repeating pattern that occurs in the time series generated by the process to be analyzed. There are few results that are capable of coping with variable-length motifs but they suffer from various drawbacks, which need more attention to be solved.

The existing solutions search for the most similar pair of pattern occurrences. From the data examination perspective, there are two possible approaches. First, the exact motif discovery, which guarantees that after finishing, there is no better pattern available in the time series under specific conditions. The conditions are usually the expected motif length, or the range of possible lengths. Requiring the length weakens the *discovery* claim of the algorithms. Moreover, the time complexity of the exact algorithms is lower-bounded to be quadratic [23], which motivates our focus on approximate algorithms that are better suited for huge time series.

The other approach is the approximate motif discovery, which gives the most similar motif occurrences with a certain probability. Such approaches can achieve better computational complexities, however, state of the art methods achieve complexities close to quadratic when they try to get good quality results. Most of them adopted symbolic representation as a dimensionality reduction technique. A step like this greatly reduces the time complexity by a constant factor by sacrificing a certain amount of data information. Approaches of random projection showed promising speedups. All the state of the art algorithms aim to improve the candidates comparison stage or data preparation stage, while we aim at reducing the required number of motif candidates, easily avoiding unpromising comparisons.

In time series where motifs are scarce, most of the comparisons in motif search are wasted on comparing sub-sequences that can be considered noise. We utilized the Birthday problem phenomenon for modeling and exploiting the behavior of standard comparison-based motif discovery algorithms. By introducing a cheap motif candidates generation strategy, we develop a motif discovery system with implicit motif length determination and any-time algorithm property. Our system is designed to keep $\mathcal{O}(N)$ time complexity for time series of $N \approx 10^9$ samples. Additionally, since most of the literature focuses on the time complexity, paying less attention to the quality measure, we propose metrics for evaluating the quality of motif discovery.

4.1.1 Notation

The following list describes the notation used for the individual variables. Bold font of small letters is used for vectors, and bold capitals are matrices. The colon is used for a range of indices forming a column vector. Let's consider a **time series** $\mathcal{T}$ as a sequence of observations:

$$\mathcal{T} = \mathbf{x}_{0:N} = [x_0, x_1, x_2, \dots, x_N]^\top , \tag{4.1}$$

where $x_t \in \mathcal{R}$ is t-th one-dimensional measurement ordered in time, N is the length of the time series. w is the length of the minimal window used for feature calculations.

The subsequence vector $\mathbf{x}_{i:i+L-1}$ is a continuous segment of time series $\mathcal{T}$ consisting of measurements x_t starting at point i and ending at point $i + L - 1$ with the length L, In this chapter, we restrict subsequences to have $L_{min} < L \ll N$, and $1 \leq i \leq N - L + 1$. L_{min} is the minimal motif size.

The motif is a subsequence $\mathbf{x}_{i:i+L-1}$, for which there is at least one more subsequence $\mathbf{x}_{j:j+L-1}$ in time series $\mathcal{T}$, whose distance (similarity) computed by the distance function

$$D(\mathbf{x}_{i:i+L-1}, \mathbf{x}_{j:j+L-1})$$

satisfy the threshold γ. The distance function have to be suitably length-normalized. Informally, motif is a segment in time series that occurs at least twice. The occurrences have to be similar.

The trivial motif is a motif whose occurrences overlap. Formally, the overlap of subsequences happens when $|j - i| < L$. Generally, the trivial motifs are unwanted discoveries.

The notation is summarized in Figure 4.1, which contains two motif occurrences of length L and one example of trivial motif.

4.1.2 Related Work

The first part of the motif discovery algorithm is data preprocessing, the main goal of which is to drop or manipulate the data before it is used to enhance and secure the performance [26]. Generally, this can be understood as a form of dimension-reduction technique.

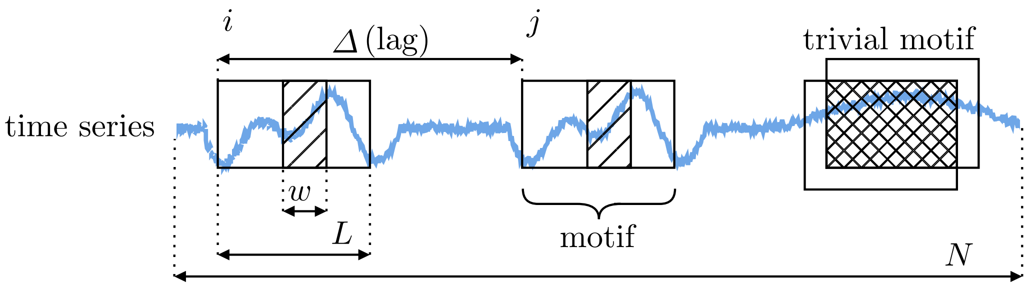


Figure 4.1: Subsequence positional notation

One of the most commonly used methods is the representation of data using the symbolic library, such as SAX (Symbolic Aggregate approXimation) and its variants. The SAX algorithm, first introduced by [27], reduces the length of time series by the PAA (piece-wise aggregate approximation) and then quantizes the continuous values resulting in a string of symbols. The SAX approach also needs to address the data granularity discovery problem. The topic of granularity discovery is discussed in [28], where the authors determine the quantization of continuous values in large databases using hierarchical clustering methods. After the conversion of time series into symbolic sequences, methods for finding the longest common subsequences can be applied. An interesting approach of byte-pair encoding was proposed in [29], where the string is gradually compressed by substituting the most common symbol pair by a new symbol. The detection of the motif is then simplified to finding the same symbol and decompressing it back to the original subsequence. This method relies on the smoothing property of SAX and degrades in performance when SAX fails to conserve important motif-related information from the original time series due to the quantization. Some authors, e.g., [30]–[32] use only the PCA for dimension reduction, others use autoencoders [33].

For the comparison of possible motifs candidates, a suitable metric is needed. It is possible to divide metrics into two main categories. The first category uses the distance-based metric. Mostly lp-norms (e.g. Euclidian) and DTW (Dynamic Time Warping) distance-based methods are used [34]. Lp-norms suffer from weak flexibility, such as a fixed length of time series. This drawback is addressed in [35], where the authors propose to normalize the Euclidean distance by the square root of the motif length to increase the robustness of Lp-norm with respect to the variable length. The DTW method is used more often than Lp-norms as it provides better results when the number of motif occurrences is small. DTW was first introduced by [36], and this algorithm allows a comparison of time series with different lengths. Note that DTW is known for its computational complexity, but this disadvantage is diminished by increasing computational resources. One of the oldest metrics used for the comparison is based on the discrete Fourier transformation (DFT) [37] and discrete wavelet transformation (DWT) [38]. It was shown that both methods have similar performance [39]. The authors in [40], [41] base their similarity metric on the histogram of gradients. Cepstrum of the time series (signal) [42], [43] can also be used. Another method uses the envelope (wave) of the signal. The envelope is a smooth curve outlining the extremes of the signal (time series). This approach helps in better matching such signals [44].

The next aspect in time-series motif discovery is selecting the optimal length of motifs. The fixed-length approach is widely used in several studies such as [45]–[47]. It examines every possible combination of subsequences of given length by a brute-force approach. The time complexity of such an approach renders this method infeasible for larger time series. A pruned variable-length motif search by enumeration was proposed by [48]. Any of these approaches brings a high time-complexity burden.

The MK (Mueen-Keogh) algorithm [23] presents an improved brute-force motif discovery.

They randomly segment the time series and then compute the distance between randomly chosen segments and the rest of the time-series segments to determine the upper bound for the motifs distances. Exceeding the bound results in early abandoning of the motif candidate, saving computational resources. This approach, which guarantees the accuracy of the motif discovery, mentions the Birthday paradox briefly as an explanation for the performance boost by the early abandoning principle.

To mitigate the time complexity, approximate motif discovery techniques emerged. Authors in [49], [50] propose the random projection algorithm that reduces the length of time series by random sampling, comparing only a fraction of the original samples. This type of sampling scheme loses a lot of information. Authors in [51], [52] proposed an approximate variable-length motif discovery algorithm that can process big time series. Their hierarchical motif enumeration algorithm utilizes a fast numerosity reduction scheme [53] and recurrent SAX word enumeration to greedily search through the vast search space. While this approach is fast, it relies on SAX transformation of the time series and thus risks missing some fine-detailed motifs.

In [54], the authors present the CHIME (Collaborative HIerarchy based Motif Enumeration) algorithm. This algorithm has three major parts. It represents the time-series using SAX as a sequence of symbols. Then it creates a tree structure to accelerate the search for the nearest neighbors. After that, using the Collaborative Enumerator, it compares two sequences in time series in one dimension. If there is a match, the algorithm looks into other dimensions to detect if the match is also in a different dimension. This approach, however, relies heavily on the SAX transformation.

In [55], the authors exploit dimensionality reduction by feature extraction. After that, they search for motifs in the continuous feature space using distance measure on the feature vectors while also optimizing the length of the motif. It is a type of so-called any-time algorithm. However, this approach requires tuning at least five parameters, making it difficult to deploy in practice.

In [56], the authors use an adaptive collision table for exact motif discovery which has a linear time-complexity in expectation. This approach needs to know the motif length and uses lp-norm as the distance function. The linear time complexity of the algorithm is based on the assumption of having continuously-valued time series, which is not the case in practice. Due to the limitations coming from quantization, their approach turns out to be quadratic in reality. We provide an analysis of the quantization effect further in Section 4.2.5.

For further details, surveys [25], [57] provide an overview of motif discovery methods, comparing them with respect to their computational complexity and robustness.

4.1.3 Contributions and Objectives

The main contribution lies in creating a Motif Discovery Framework (MoDiF), where a modular approach to motif discovery using approximation methods is used. The modularity brings independence of other typical motif discovery steps such as similarity measure, symbolic representation of time series, motif-length determination, etc. The ability of our approach to reduce the number of motif candidates makes it an excellent asset to be incorporated into any motif-discovery pipeline. It possesses the any-time property. Moreover, thanks to the framework's modularity, our approach does not impede other steps in the motif-discovery pipeline.

There are many *exact* motif discovery algorithms, but their time complexity is high and cannot be lowered under $\mathcal{O}(N^2)$ (see [23]). Approximation methods overcome the exact-methods complexity issue, but as described in Section 4.1.2 there are some disadvantages. The approach presented in this chapter is unique and tackles an aspect of the motif-discovery problem which has not been exploited up to now. This aspect concerns the pre-filtration of the time-series data and we show an efficient way to combine it with virtually any other motif-discovery method.

The objective of this chapter is to show how the contribution was achieved and emphasize metrics of motifs quality, which is usually of low interest in motif-discovery literature.

We start with problem formalization in Section 4.2.1 and a high-level overview of the MoDiF in 4.2.2. Then, in Section 4.2.3, we describe the details of how the feature vector is created to access the time-series subsequences efficiently. In Section 4.2.4, we analyze the birthday paradox and in Section 4.2.5 focus on how to set parameters to reduce the growth rate of the number of equal pairs. This leads to the main contribution of the chapter; how to use and set up the collision table to discover motifs efficiently. This contribution is further elaborated in Section 4.2.6, where it is applied to the motif-length search task. The formal complexity analysis of the framework is provided in Section 4.2.7. The theoretical contribution is evaluated by experiments in Section 4.3. The results are discussed and future steps are outlined in Section 4.4.

4.2 Motif Discovery Framework – MoDiF

4.2.1 Problem Definition

Given a univariate time series $\mathcal{T}$, we search for a maximally covering set of motifs ranked by their similarity. The minimal length of a motif is set to be L_{min}, so any shorter motif candidates are disregarded. Motifs can be of any length, but we restrict our problem to no time-warping. Motifs are assumed to be noisy and the algorithm must run in the any-time regime.

We assume the time series to be a stationary random process with normally distributed values, which can be achieved by removing trends and normalization during data preprocessing. This is a widely adopted empiric assumption [58], [59].

4.2.2 High-level Architecture

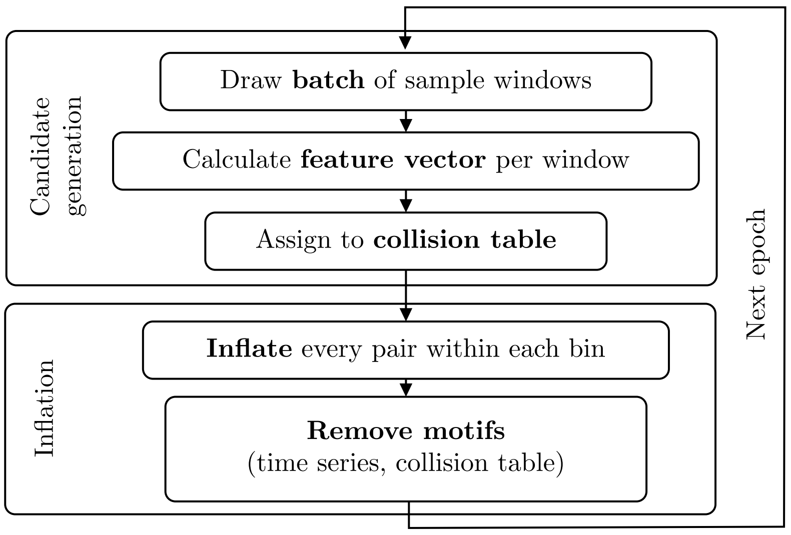
Here, we describe the high-level architecture of MoDiF. It has two major phases, as depicted in Figure 4.2a – the *candidate-generation* phase and the *inflation* phase. These two phases alternate until all samples are examined or an early termination is invoked. We denote a single cycle of the two phases as an epoch.

We start with the idea that any motif consists of subsequences that are motifs themselves. The shorter subsequences are cheaper to evaluate for similarity. We take the extreme step of searching for the shortest possible subsequences that are similar to each other. We denote such short subsequences as *windows*.

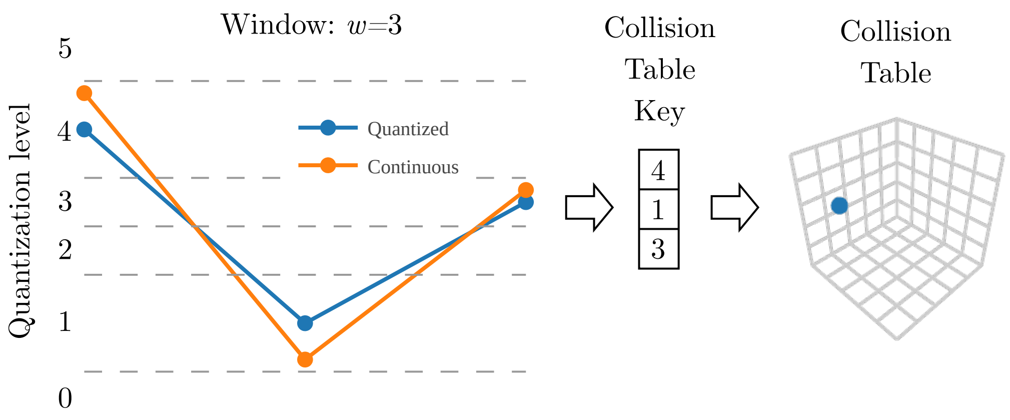
In the candidate-generation phase, we pick windows from the time series randomly without repetition and store them in the *collision table*. The collision table is an associative array (also called dictionary) consisting of *(key, value) pairs*, where the *value* is a bin containing references of the chosen windows in the time series, and the *key* is a feature vector (see Section 4.2.3). An illustrative example of the binning procedure is in Figure 4.2b. If the feature vector of two or more windows is the same, these windows are similar, which is indicated by a collision. In the dictionary, the references to the colliding windows are stored in a bin under the same key. The chance of collision grows rapidly with the number of draws because of so-called birthday paradox, as explained later in Section 4.2.4. The motif candidates are generated in batches to improve computational efficiency.

The second phase takes the motif candidates from the collision table and searches for optimal similarity by varying the length of motif candidates as described in Section 4.2.6. We call it the *inflation* phase. If the motif fails to overcome a pre-defined threshold of the similarity metric, it is discarded. If the motif is accepted, its occurrence is removed from the time series. The next epoch continues the search only in the remainder of the time series. All the samples covered by the discovered motifs are also removed from the collision table.

Computationally, the inflation phase is orders of magnitude more expensive than the candidate-generation phase. The key idea here is to reduce the number of candidates coming from the candidate-generation phase. Only pairs that overcome the threshold, pass the first filter.


Candidate generation
Draw **batch** of sample windows
Calculate **feature vector** per window
Assign to **collision table**
Inflation
Inflate every pair within each bin
Remove motifs (time series, collision table)
Next epoch

(a) Motif Discovery pipeline design



(b) Window quantization and binning procedure example

Figure 4.2: Algorithm procedure diagrams

4.2.3 Feature Vector

The choice of features has a significant impact on the properties of the proposed motif candidates. We chose the window itself as a feature vector which is the most efficient computational solution. Each w-long window becomes feature vector $\mathbf{v}_i = \mathbf{x}_{i:i+w-1}$, where i is the index to the time series $\mathcal{T}$. The length w affects the localization precision of the algorithm. In practice w ranges from 3 to 20, determined by the strategy described in Section 4.2.5. The feature vector needs to be quantized because the quantization allows for exact equality of the feature vectors which indicates the similarity of the windows. The feature vectors need to be uniformly distributed across the bins in the collision table, which is achieved by uneven quantization. Intuitively, if all bins contain approximately the same number of references to the windows in the time series, the computational efficiency is the best. The uneven quantization thresholds are calculated as follows. Assuming the feature vector is jointly normally distributed random vector, we apply the whitening transform [60] on $\mathbf{v}_i$:

$$\mathbf{z}_i = \mathbf{W}\mathbf{v}_i, \mathbf{W} \in \mathbb{R}^{w\times w}, \tag{4.2}$$

where the whitening matrix $\mathbf{W}$ is chosen so that the following condition is met:

$$\begin{aligned} \mathbf{W}\mathbf{W}^{\top} &= \mathbf{\Sigma}^{-1}, \mathbf{\Sigma} \in \mathbb{R}^{w \times w} \\ \mathbf{\Sigma} &= cov(\mathbf{v}_i, \mathbf{v}_i) \end{aligned} \tag{4.3}$$

where $\mathbf{\Sigma}$ is the covariance matrix of $\mathbf{v}_i$. We use the unbiased estimate of the covariance:

$$\mathbf{\Sigma} \approx \frac{1}{|\mathcal{V}| - 1} \sum_{\mathbf{v}_i \in \mathcal{V}} (\mathbf{v}_i - \boldsymbol{\mu}_{\mathbf{v}})(\mathbf{v}_i - \boldsymbol{\mu}_{\mathbf{v}})^T,$$

where $\boldsymbol{\mu}_{\mathbf{v}}$ is the sample mean vector of $\mathbf{v}$, $\mathcal{V}$ is the set of Monte Carlo sampled feature vectors from the time series, and $|\mathcal{V}|$ the size of the set $\mathcal{V}$. The matrix decomposition (4.3) is obtained, e.g., by the Cholesky decomposition or by the singular value decomposition. We apply the quantile transform on the whitened vector $\mathbf{z}_i$, which approximates the probability integral transform [61]. In practice, assuming jointly normal distribution of feature vector holds well [59]. Note that the quantile transform spreads out the denser regions and concentrates the sparse regions and can be applied on arbitrarily distributed features. Thanks to that, the transformed feature vectors become approximately uniformly distributed.

The whitening procedure can also be viewed as an application of series of linear filters that extract a feature vector from the window $\mathbf{x}_{i:i+w-1}$ features are independent, i.e., orthogonal. For example, band-passing filters with non-overlapping bands satisfy the property of orthogonality.

The total bin count plays a key role in the reduction ratio. It will turn out in Section 4.2.7 that in the worst case scenario when all $N - w + 1$ windows have to be examined, the expected number of compared pair candidates is reduced by a factor exponential in $\dim(\mathbf{v}_i)$. This is closely related to the birthday paradox as described in the next section.

4.2.4 Birthday Paradox

In brief, the birthday paradox states that a group of 23 randomly chosen people have more than 50 % chance to contain at least one pair with the same birthday. On the other hand, the probability of someone having birthday on a particular selected day of the year is only $1 - (364/365)^{23} \approx 6.1\,\%$. The difference in probability is counter intuitive and thus somewhat paradoxical. The rate of growth of the probability of any birthday collision is very high and surpasses 99 % in the group of 60 people.

This surprisingly quickly growing probability of collision manifests itself in the candidate-generation phase of our system. As shown in Section 4.2.3, the feature vectors are supposed to be distributed uniformly over the collision table. Let the total number of bins in the collision table be denoted as d and expressed as

$$d = b^{\dim(\mathbf{v}_i)}, \tag{4.4}$$

where b is the number of bins in a single dimension of the collision table. Then the probability $p(s; d)$ of at least one collision in the collision table with the total of d bins when s windows are drawn (independently of each other) is

$$\begin{aligned} p(s; d) &= 1 - \prod_{i=1}^{s-1} \left(1 - \frac{i}{d}\right) && (4.5) \\ &\approx 1 - \exp\left(-\frac{s(s-1)}{2d}\right) && (4.6) \\ &\approx 1 - \left(\frac{d-1}{d}\right)^{\dfrac{s(s-1)}{2}}, && (4.7) \end{aligned}$$

given $s \leq d$. If $s > d$, then $p(s; d) = 1$. The random draw of windows is required to be done without repetition. Equations (4.5)-(4.7) depict how quickly the probability of collision grows.

The expected number of colliding windows N_c when s windows were drawn, is critical, since we compare only the N_c colliding candidate pairs. We will derive it from the binomial distribution. From the perspective of a bin, the binning process is a repeated Bernoulli process with probability of success (hitting the current bin) $1/d$. The probability of a particular number of successes (hits) k of a single bin follows the binomial distribution $\mathcal{B}(k; s, 1/d)$, where the number of trials is s and the probability of success is $1/d$. The expected number of all bins with exactly k hits denoted as N_k is expressed as:

$$N_k(k) = \sum_{i=1}^{d} \mathcal{B}(k; s, 1/d) = d\mathcal{B}(k; s, 1/d) = d\binom{s}{k}\left(\frac{1}{d}\right)^k\left(1 - \frac{1}{d}\right)^{s-k}, \tag{4.8}$$

where $\mathcal{B}(k; s, 1/d)$ is the probability mass function of the binomial distribution.

For the evaluation purposes, the expected number of bins containing at least m hits can be expressed from the cumulative density function (cdf) of the binomial distribution as:

$$N_m = d\left(1 - \mathrm{cdf}_{\mathcal{B}(m-1; s, 1/d)}\right), \tag{4.9}$$

where $\mathrm{cdf}_{\mathcal{B}(m-1;s,1/d)}$ stands for the binomial distribution cumulative density function of at most $m - 1$ successes given s trials with trial success probability $1/d$.

The expected number of all candidate pairs N_c generated from the collisions is the sum of expected combinations using (4.8):

$$N_c = \sum_{i=2}^{s} \binom{i}{2} N_k(i). \tag{4.10}$$

We tested the validity of models (4.8), (4.9), and (4.10) on a simulated white-noise time series. In Figure 4.3a, the number of expected candidate pairs from (4.10) are compared to the observed number in the experiments. The pair plot in Figure 4.3c depicts how the whitened feature vectors are evenly distributed in the collision table, which illustrates the effect of the quantile transform.

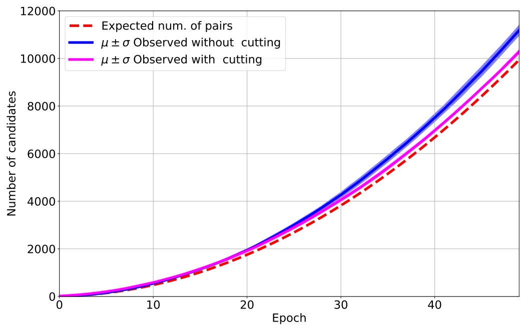
Uniformly distributed windows will generate the least number of collisions in the collision table. Any time series containing motifs generates more collisions. This notion justifies the use of the collision table as a means to find the motifs efficiently. To give the motifs chance to collide before the system is saturated by random collisions, we need to set up parameters of the collision table b and $\dim(\mathbf{v}_i)$ to keep the expected number of candidates N_c manageable for all s. We propose a method for setting the parameters in Section 4.2.5.

4.2.5 Collision Table Parameters

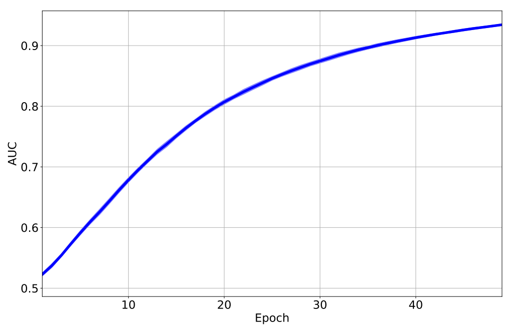
The success of our approach lies in a correct design of the collision table. This subsection analyses the key cornerstones for setting the right parameters.

When a new collision occurs in a bin with more than one window, as many new candidate pairs are generated as there are windows already in the bin. As soon as $s > d$, multiple windows in at least one bin are inevitable (due to the pigeonhole principle), and the algorithm turns to have time complexity of $\mathcal{O}(s^2)$. That is why we need to keep d big. Luckily, the collision table size defined in (4.4), grows exponentially with the exponent $w = \dim(\mathbf{v}_i)$. We need to adjust the bin count d so that the number of candidate pairs modeled by (4.10) is minimized. This task can be solved iteratively resulting in a desired value of d, but a trivial solution is to choose d as big as possible.

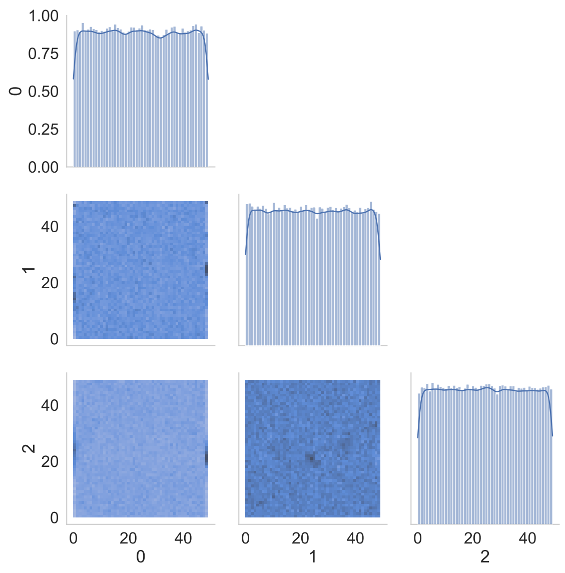
To increase d, we can either increase b or w, but there is a limit to increasing either of those. For b, we are limited by the quantization and the finite representation of numbers in



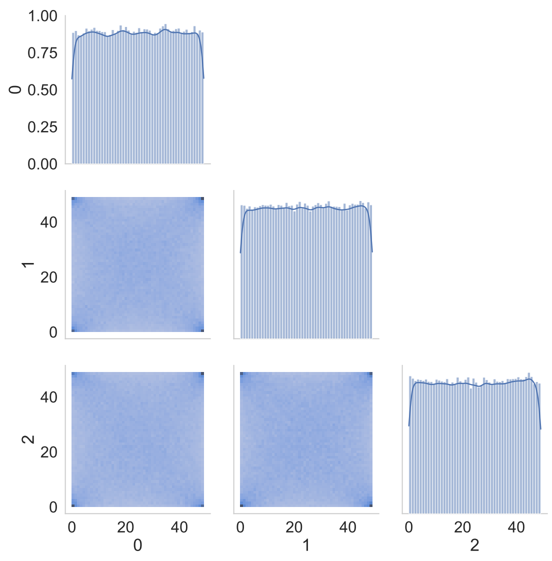
(a) Predicted and observed number of candidates on the Artificial data set of *cover* $= 0.28\,\%$



(b) AUC evolution ($\mu \pm \sigma$) during epochs on the Artificial data set of *cover* $= 0.28\,\%$



(c) Collision table occupancy for the Artificial data set of *cover* $= 0.28\,\%$



(d) Collision table occupancy for the ASTRO data set [62]

Figure 4.3: Behavior of the MoDiF for $b = 50$, $w = 3$, $batch = 1000$, 10 experiment repetitions.

digital computers. After a certain point, increasing the number of quantization levels b will not divide windows into different collision-table bins anymore. Another aspect is that too small quantization levels may intensify the problem of weak motifs. A weak motif is motif that is not an exact copy of its template but rather its perturbation. When a symbolic representation in preprocessing is applied (e.g., SAX), the resolution in values is lowered, but the probability of weak motifs is lowered too as the neighboring values are likely to be projected to the same symbol. Weak motifs are less likely to produce an exact hit in the collision table. Suppose $\tilde{x}_i$ is a discrete (quantized) value of x_i. Also suppose, we have a probability p_{ql} of x_i not missing its true quantization level $\tilde{x}_i$ due to noise, then the probability of the whole window of length w not missing a single sample is

$$p(\tilde{\mathbf{x}}_{i:i+w-1} = \tilde{\mathbf{x}}_{j:j+w-1}) = \prod_{k=0}^{w-1} p(\tilde{x}_{i+k} = \tilde{x}_{j+k}) = p_{ql}^{w}. \tag{4.11}$$

For $p_{ql} = 0.95$ and $w = 10$, the probability is approximately 0.6, whereas for $p_{ql} = 0.99$ and $w = 10$, the probability is more than 0.9. This notion underlines the importance of noise-suppressing preprocessing of the time series.

We developed a strategy for tuning the parameter b in the following way. First, identify the noise standard deviation and set the smallest quantization level to be three times bigger than that. This reduces the probability of missing a correct quantization level due to noise under 0.03. Then, we set the probability h of w consecutive successful hits to an acceptable value, e.g., 0.9. Formally:

$$p_{ql}^{w} > h = 0.9 \tag{4.12}$$

The window length w is then upper bound by solving the equation (4.12) for w by:

$$w \leq \frac{\log h}{\log p_{ql}} = \frac{\log 0.9}{\log 0.97} \approx 3.45. \tag{4.13}$$

The equation (4.13) gives a guidance towards a reasonable choice of w that comes only from an analysis of noise level contained in the time series data.

Note that when using a sparse data structure of the hash table, the memory complexity is $\mathcal{O}(N)$, independent of the number of the quantization levels while the expected access time remains $\mathcal{O}(1)$.

The collision table have to be sufficiently big, but also the distribution of hits in it have to be as uniform as possible. We wish to compare more frequently such collisions that originate from occurrence of a motif. These notions motivate to transform the feature vectors so that they have multivariate uniform distribution and only statistical outliers – such as frequently repeated patterns (motifs) – cause local spikes in the concentration of otherwise uniform collision table. This justifies the application of the whitening transform in (4.2) and the quantile transform described in Section 4.2.3.

4.2.6 Inflation Phase

When a collision-table bin has more than one window, i.e., a collision occurs, and the new candidates are passed for further examination. Such examination runs resource-heavy metric to verify or discard the candidate. To verify a candidate pair, we search for the motif length and optimal lag. Formally, we define this task as minimizing the length-normalized distance of the candidate pairs:

$$\begin{aligned} \min_{i,L} \quad & D(\mathbf{x}_{i:i+L-1}, \mathbf{x}_{i+\Delta:i+\Delta+L-1}), \\ \text{s.t.:} \quad & L_{min} < L < L_{max} < N/2, \end{aligned} \tag{4.14}$$

where Δ is the lag between motif's occurrences. i is the beginning of the first occurrence, the other occurrence begins at index $i + \Delta$, and both occurrences end L samples from their beginnings. As discussed, e.g., in [51], the normalized distance is often unstable for small L so we limit the examined lengths L to be at least L_{min} long. Reasonably small value of L_{max} significantly accelerates the search.

The distance function D can be chosen arbitrarily, and this choice has influence on what kind of motifs can be discovered. This distance function have to adapt to the variable length of motif candidates. In [35], the authors propose to normalize the Euclidean distance by the candidate length square root. Formally, the length-normalized Euclidean distance D_E is computed as

$$D_E(\mathbf{y}, \mathbf{z}) = \frac{1}{\sqrt{L}} ||\mathbf{y} - \mathbf{z}||_2, \tag{4.15}$$

where $\mathbf{y}, \mathbf{z}$ are equally long subsequence vectors, L is the length of the subsequences, and $||.||_2$ denotes the 2-norm of the vector. This metric can be computed very efficiently by utilizing the summed-area tables allowing for a fast optimization of (4.14). The details of this optimization are out of the scope of this paper, which aims to be independent of the selected distance function.

To further accelerate the search, the discovered motif occurrences are immediately removed from the time series if approved. The motif is approved, if the candidate pair have distance smaller than a predefined threshold, which is the average distance of randomly selected combination of segment pairs from the time series. This approach was studied in [35]. This is a greedy search, but it accelerates the search significantly and experiments in Section 4.3 show that the quality metrics of motif discovery are not significantly affected.

4.2.7 Complexity

In the worst case, all $N - w + 1$ windows have to be visited and placed into the collision table in the candidate-pairs generation phase. Both the computational and the memory complexity of this process are $\mathcal{O}(N)$. The random access to the collision table has complexity $\mathcal{O}(1)$ since associative array is used as shown in Section 4.2.2. Thanks to the quantile transform, which is used to balance the occupancy of the collision table, each bin is expected to contain the same number of windows. That is, each bin contains on average $(N - w + 1)/b^w$ windows.

In the inflation phase, all combinations of pairs from each bin have to be evaluated, i.e., inflated and approved or disapproved. That represents the computational complexity of $\mathcal{O}(N^2/b^{2w})$. This step reduces the complexity by a rapidly growing factor. This is where the approved-motifs removal significantly prunes the search space, especially in the time series, where the motifs are abundant. The strategy for choosing b and w is designed so that the factor $1/b^w$ is always growing faster than N^2 resulting in the linear time complexity $\mathcal{O}(N)$. The linear complexity then holds as long as the motifs have length at least w, which is met in most cases in practice. This is a great improvement compared to exact motif-discovery methods, where the lower bound of the complexity is $\mathcal{O}(N^2)$ (see [23]).

4.3 Experiments

In the following section, the key properties of our algorithm are experimentally examined. There are two main algorithmic properties of interest, the **qualitative** performance and the **speed** performance. The qualitative performance measures how similar the occurrences of a discovered motif are and what portion of input data are covered by the occurrences. The speed performance describes the practical computational demands of the algorithm. The speed is measured as the total time of processing benchmark data sets.

All experiments were performed on a 2021 MacBook Pro with an M1 Pro chip (10-core CPU, 16-core GPU) and 16GB of memory.

4.3.1 Applied Methods and Metrics

Assessing the quality of a discovered motif is a complicated task and is sparsely investigated in the literature. The main reason is that motifs are by definition unknown before they are discovered. We can compare discovered motifs on artificially composed data sets – the Implanted motifs data sets [63]. In such data sets, we know which samples belong to the motifs and which lie outside.

Under these assumptions, we can utilize the classification metric of the false discovery rate (FDR) defined as:

$$\mathrm{FDR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TP}}, \tag{4.16}$$

where TP is the number of true positives, i.e., samples from motifs occurrences correctly labeled as belonging to a motif, FP is the number of false positives, i.e., samples outside motifs occurrences incorrectly labeled as motif samples. The FDR depends on the decision threshold, which needs to be chosen specifically for the data set under examination. The FDR metric is between 0 and 1, where smaller is better. A value of 0 means that all labeled samples belong to a motif, and one means that all labeled samples are outside any of the motifs.

Another well established metric suitable for binary classification tasks is the area under the receiver operating characteristic (ROC) curve, i.e., AUC metric. For this metric, we labeled the motif samples by the best achieved similarity. That means when an algorithm proposes overlapping motifs, the overlapping samples get the best similarity of the propositions. The AUC metric, which is classification-threshold-invariant, takes values between 0 and 1, where 1 is the best performance and 0 is the worst case. For more details on these metrics, see, e.g., [64]. Indeed, to evaluate FDR or AUC, we need to know the ground truth about all samples – whether they truly belong to a motif or not, and this information is unavailable for most real-life data sets.

Metrics calculated on simulated data sets bring only limited insights into the performance of algorithms on real-life data sets. For the real-life data sets, we propose the motif coverage metric for qualitative comparison of results. It is defined as follows.

$$cover(\mathcal{T}, L_{min}, D, \gamma) = \frac{|\mathcal{M}|}{N} = \frac{\mathrm{TP} + \mathrm{FP}}{N}, \tag{4.17}$$

where $\mathcal{M}$ is the set of all samples x_i that are contained in at least one motif occurrence and N is the length of the time series $\mathcal{T}$. The *cover* depends on the given time series $\mathcal{T}$, lower bound on motif length L_{min}, distance function D, and the threshold γ lower-bounding the distance between motif occurrences. The best value of *cover* depends on the time series and it is empirically estimated for the selected real life data sets listed in Table 4.2 in column $cover_{\mathbb{E}}$. The further the *cover* from the estimated value is the worse the performance of motif discovery algorithm. The opposite cannot be said with certainty as the ratio between TP and FP in (4.17) are unknown. Even exactly the same coverage as the expected one does not guarantee discovering all the motifs in data set. At the same time, a very different coverage from the expected one guarantees a lot of false motif discoveries. The problem is in determining the expected coverage. The values of *cover* are between 0 and 1. Note that the motif overlaps are counted only once.

4.3.2 Experiments on Implanted Motifs Data Sets

The ground truth of motifs location is unavailable for most of the real-life data sets. Therefore, we prepared data sets with motifs implanted into a Wiener-process-generated time series. The Coffee [65] data set was used as a source of the motifs. We generated four data sets (time series) with different values of $cover_{\mathbb{E}}$. The motifs in these data sets were also corrupted by additive noise. Each of the four time series consists of 10^6 samples with $cover_{\mathbb{E}}$ ranging from 0.1 % to 29 %. We call these data sets the Artificial data sets.

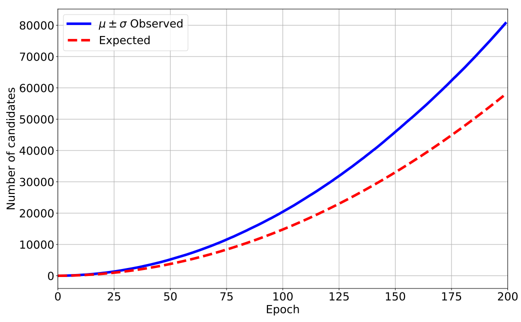
The pre-processing pipeline removes trends (by a high-pass filter), and then standardizes the time series by subtracting its mean value and scaling it by its reciprocal standard deviation. After performing these steps the time series changes from a Wiener process into a stationary stochastic process with additive white noise with zero mean and unit variance.

Most of the best-performing motif discovery algorithms utilize the PAA or SAX representation of the time series to reduce the amount of data to process, as discussed in Section 4.1.2. We adopt the PAA time-series representation for the results to be comparable with the results of these algorithms. Note that this step of performing PAA also reduces the motif discovery resolution in the time axis by aggregating time-series subsequences into single values.

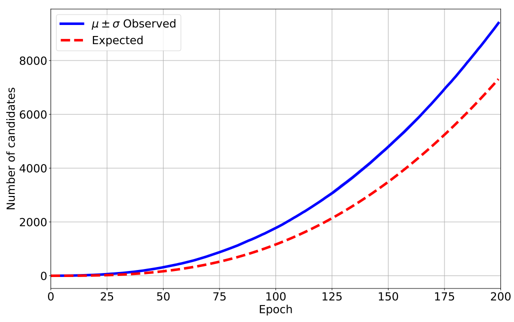
To evaluate the FDR metric, we have to select appropriate threshold for each motif discovery algorithm. In our case, we randomly select windows from the time series and we inflate them using equations (4.14) and (4.15) and calculate the mean of their similarity. Then, we generate first 100 candidates from the collision table and inflate them according to the same equations, and calculate their mean similarity as well. The threshold is set to the middle between the mean of the random-subsequences similarity and the mean of the 100-candidates similarity. For CHIME, we use the threshold provided by its authors in [54]. The SCRIMP++ authors do not provide a single similarity threshold to extract motifs. They simply provide the single most similar pair. To extend the SCRIMP++ to our experimental settings, we provide the optimal threshold set to the n-th best metric provided by the SCRIMP++ algorithm. The number n is the number of the implanted motifs in the data set under examination. This step brings information about the ground truth to the SCRIMP++ algorithm, so its performance metrics are improved as indicated in 4.1. However, such an improvement is impossible for real-life data sets, where the ground truth is unavailable.

We use the any-time property of our algorithm to terminate after the first 1/4 of epochs. To support the idea of early stopping, we examined the evolution of AUC metric depending on the number of epochs, which is shown in Figure 4.3b. The AUC metric exceeds 0.9 after 38 epochs. This behavior was observed in every Artificial data set.

Table 4.1 shows performance comparison of the motif discovery algorithms. Our algorithm and CHIME both perform the discovery of variable-length motifs. Their speed is comparable and so is the AUC metric but the FDR metric is several orders of magnitude worse than our algorithm. SCRIMP++ can discover only fixed-lenght motifs. This disadvantage can be solved by running the algorithm 985 times (motif lengths from 15 to 1000). This fact significantly prolongs the time needed to obtain the results from SCRIMP++, e.g., for $cover_{\mathbb{E}} = 0.286\,\%$ the computation time would be 66 438 s instead of 67.45 s show in Table 4.1. SCRIMP++ has almost perfect AUC, because the algorithm was provided with the exact motif-length information, which is unavailable for the real-life data sets. We must be aware that our algorithm reflects real-life situations (variable motif lengths etc.) while SCRIMP++ does not.



(a) Expected and observed number of candidates



(b) Number of bins containing exactly 3 candidates

Figure 4.4: Performance of our algorithm on ASTRO [62] data set; $batch = 1000$, $w = 3$, $b = 70$, 10 experiment repetitions.

Algorithm:	**Our**			CHIME			SCRIMP++		
$cover_{\mathbb{E}}$	AUC ↑	FDR ↓	Time ↓	AUC ↑	FDR ↓	Time ↓	AUC ↑	FDR ↓	Time ↓
.00286	.883	.006	148.89	.950	.999	150.39	1.000	.026	67.45
.01001	.738	.011	148.82	.478	.991	137.94	1.000	.016	67.43
.0286	.818	.006	157.50	.973	.973	130.51	1.000	.012	68.39
.286	.934	.008	250.97	.572	.713	269.85	.996	.015	72.79

Table 4.1: Qualitative comparison of algorithms on Artificial data set [63] based on [65]. The arrows in label row indicate whether a bigger (↑) or smaller (↓) value is desired ($batch = 1000$, $b = 50$, $w = 3$). Our algorithm terminates after the first 1/4 of epochs. For CHIME and Our algorithm, the motif length search was from 15 to 1000. SCRIMP++ examines only a single motif length, for variable length, repeated restarts are necessary so that e.g. for $cover_{\mathbb{E}} = 0.286\,\%$ the computation time would be 66 438 s instead of 67.45 s.

4.3.3 Experiments with Real-Life Data Sets

The experiments on real-life data sets show that the assumptions of normally distributed data are difficult to hold, which manifests in higher-than-expected number of candidates as shown in Figure 4.4a. The more a data set differs from normally distributed data, the less effective the quantile transform is. In turn, the occupancy of the collision table is more concentrated in clusters of frequent data, as shown in Figure 4.3d. This leads to a faster generation of motif candidates than estimated by theory in Section 4.2.4, but the divergence of expectations and observations is not too radical when $cover_{\mathbb{E}}$ is low, see Figures 4.4a and 4.4b for ASTRO data set.

The experiments on the real-life data sets are listed in Table 4.2. The thresholds from Section 4.3.2 were also used for the real-life data sets. For data sets with a lower motif frequency corresponding to a lower $cover_{\mathbb{E}}$, our algorithm is faster than the others. The *cover* metric of our algorithm closely approaches the estimated $cover_{\mathbb{E}}$. CHIME algorithm shows very high *cover* for all data sets, even for the ASTRO data set, in which motifs are very rare. Our algorithm proposes only very likely motif candidates, which is the desired behavior. This is encouraged by the smart pre-filtering of candidates, which intensifies the probability of motifs being examined early. The discovered motifs are then cut out which further reduces the data set length.

Algorithm:		**Our**		CHIME		SCRIMP++	
Data set	$cover_{\mathbb{E}}$	*cover*	Time [s] ↓	*cover*	Time [s] ↓	*cover*	Time [s] ↓
ECG [66]	.97	.986	358.17	.999	132.77	.757	62.167
GAP [67]	.95	.95	536.82	.97	124.36	.153	60.145
ASTRO [62]	.001	.032	278.31	.98	817.49	.271	58.004
Robot [68]	.373	.387	29.12	.99	132.03	.828	58.604
Oven [68]	.12	.091	6.77	.995	.50	.113	.153

Table 4.2: Qualitative comparison of the motif-discovery algorithms on real life data sets. Column $cover_{\mathbb{E}}$ is the cover expected in data set. The arrows in label row indicate whether a bigger (↑) or smaller (↓) value is desired ($batch = 1000$, $b = 70$, $w = 3$). Window length limits were set to $L_{min} = 15$, $L_{max} = 1000$.

4.3.4 Industrial Use Cases

MoDiF was used on power-consumption data collected in a car-manufacturer robotic cell. The time series consisted of 1 million samples. The robots were performing repetitive operations to weld and glue parts, and these operations projected to the robots power consumption as motifs. MoDiF was used to prepare the data to train machine learning models. Thanks to using MoDiF, the data set preparation process was fully automatic without any manual intervention. Our framework proved to be well suited for this task as it serves as a whole data-set preparation pipeline.

Apart from the industrial robot use case, MoDiF was also deployed on another industrial use case of batch-heating ovens, where the motifs were used for an accurate estimation of the availability of the oven, which requires a really low FDR. The length of this time series was 35 thousand samples. The results of both these use cases are listed in Table 4.2.

4.4 Discussion and Results

This chapter explores the impact of a collision table utilized as a motif-candidates pre-filtering technique. To our best knowledge, it is the first work that investigates thoroughly the elemental phenomena driving the effectiveness of the collision-table pre-filtering and its intentional application for motif discovery.

The problem of quadratic time complexity with most of the current motif discovery algorithms is usually tackled from two distinctive viewpoints. The first viewpoint is a deep analysis of the similarity metrics. By bounding a chosen similarity metric, using an early-abandoning technique or a dynamic-programming technique, it is possible to achieve linear time complexity. The other viewpoint examines the length of data. By careful dimensionality-reduction methods (e.g., by symbolic aggregation), the length of the data is greatly reduced, however, at the cost of a resolution loss.

Our work brings up a third viewpoint, which is the pre-filtering of candidate pairs before they are compared for similarity. This approach is independent of both the dimensionality reduction and the similarity metric. It can be easily combined with most of the popular motif discovery algorithms to significantly reduce the number of similarity comparisons. Thus, the proposed method has the time-complexity of $\mathcal{O}(N)$ in expectation. The linear time complexity holds for the time series having tens of millions of samples, which is true for most real-life applications.

Also, we proposed the inflation technique to solve the problem of unknown motif length and we experimentally showed how it affects the quality of the discovered motifs. Our algorithm works in the any-time regime and requires only the minimum and maximum lengths of motifs, and optionally also the batch size, to be set. All the remaining parameters are tuned automatically. The discovered motifs are presented to the user, ranked and ordered according to length-normalized similarity.

To evaluate the quality we chose two different quality metrics. The results on artificial data sets show an excellent performance of the Motif Discovery Framework (MoDiF) presented in this chapter. The performance was further validated on real-life data sets and on data sets from two industrial use cases. The results confirmed the expectations gained from the experiments on the artificial data sets.

The supporting software code is available on GitHub [69], the data sets created for this chapter are available online at [63], [68].

Chapter 5

Clustering of Patterns

Previous chapters described methods for manual and automatic segmenting timeseries – motif discovery. The resulting segments (alias patterns) make a non-homogeneous set of objects that must be carefully partitioned to form a training data set for the target machine learning tasks. The repetitive robotic operations produce similar patterns. This property leads to the idea of clustering the patterns according to their similarity. However, this idea brings challenges, such as measuring the similarity of sequences of unequal lengths or how many clusters should be expected. We tackled these issues in paper [3], where we contributed with a similarity measure for sequences of unequal lengths. This chapter concludes the data-set preparatory phase of this dissertation. Note that we return to using the more general term *pattern* instead of *motif* in this chapter for clarity.

5.1 Introduction

A sustainable production as a part of a larger picture of Industry 4.0 begins to require energy effectiveness. Especially robotic applications have been getting higher attention recently because the robots consume about 25 % of electrical energy in highly robotized industries such as car production. There have been approaches that focus on decreasing the energy consumption of the robotic lines such as [1], [70] or [71]. The latter relies on the knowledge of the robotic operations in terms of the electrical energy that is consumed by each robot in the production cell during a specific movement. Based on the measurement of the energy consumed by each robot it is necessary to separate the time series of the sampled values into smaller segments that correspond to the robotic operations. In paper [10] an approach to identify the robotic operations was introduced which was based on the correlation with the series that were used for teaching the model. This task was similar to the non-intrusive load monitoring (NILM) in smart buildings such as in [72] except for much higher dynamics in robotic applications.

Another area of application is the diagnostics of robots based on the knowledge of their electrical energy consumption during the executed operations. If a suitable diagnostics model is created it can be taught on the measured time series of energy consumption values and then the model can be used for the robot diagnosis. Paper [3] shows a way how to create a Markov model of an industrial robot and identify the robotic operations based on their probability of resemblance to the learned trajectories.

The chapter describes an improved method to identify the robotic operations with respect to the previous work in [3] and [10], which is needed for the above described applications of energy optimization and robotic cell diagnostics. In [3] models to estimate the energy consumption of industrial articulated robots were developed. For an unsupervised learning of the models, preprocessing steps on data are required. When the robotic operation non-intrusive monitoring task was first approached in [10], an external approach for measuring similarity was used. This idea is revisited in this paper, and a modification of Pearson's correlation coefficient is used as a basis for measuring a distance between sequences. It still relies on measuring the energy consumption of the robot in a production cell.

5.1.1 Objectives

The basic problem formulation goes as follows. A robotic cell consisting of multiple robotic manipulators is non-intrusively monitored. Monitoring is done by measuring the power consumption time series of each robot, which is sampled with 40 ms sampling period and time series covering weeks of data have to be processed. Long one-dimensional data contains repeating segments, which correspond to the robotic operations. There are various operations in the data. In this chapter, we develop an unsupervised data preprocessing which segments-out repeating patterns and prepares a set of training examples for subsequent unsupervised CS-HGMM training.

In [10], the data were preprocessed manually, and a trained supervisor had to extract repeating patterns by eye. After templates had been prepared a search in the data was run and statistical analyses were conducted. These analyses are used for example for an evaluation of the actual impact of optimization of robotic operations.

In this chapter, we focus primarily on a robust training data set preparation. The segmenting step described here can be replaced by a smarter segmentation solution such as motif discovery described in Chapter 4. However, various segmentation approaches appear in practice. We want our solution to be modular and adaptable to a wide range of segmentation results. For that reason, we investigate a use-case dependent segmentation by signal-plateau detection. The goal is to cluster segments robustly considering the variability in segment boundaries.

5.1.2 Related Work

Interest in NILM has risen only in last few years in industry as the power consumption of the automated production increases. NILM methods usually rely on load signatures of the devices. A load signature is some characteristic feature of the appliance which manifests in the measured data. The signature vastly reduces the dimension of the descriptor of the power consumption of the appliance and thus makes distance measuring methods feasible in real applications. In [73] the signatures of voltage-current trajectories were used followed by their hierarchical clustering. In the work of [74] a disaggregation of load signatures based on active and reactive power consumption and then cluster them using k-means clustering method. The disadvantage of k-means is that it detects noise signatures based on their distance from its cluster centroid. The other drawback is the need to estimate the number of cluster before the k-means run.

The work of [72] targets the varying load signatures by exploiting the fact that the power to the different classes of appliances loads the power line asymmetrically. The asymmetry aids in distinction and disaggregation of the appliance signatures. The robotic cells studied in this paper are not wired this way and such requirement is difficult to fulfill in practice due to power grid balancing demands.

The authors of [75] use on-off load edges for detection of events in a monitored grid. Then they apply wavelet and Fourier analysis to extract features from data in the vicinity of the switching edge to classify active appliances. The frequency analysis showed to be unsuitable for the robotic operation detection as discussed in [10].

As for the clustering of time series data, the current trend is to adopt a particular cluster analysis for timeseries segments. In [76] authors use Gaussian mixture models to cluster the feature vectors of hundreds of elements using EM algorithm for unsupervised estimating the GMM.

An adaptation of DBSCAN with dynamic time warping as a distance function is used in [77] inspired by timeseries mining. The DTW distance assigns the same weight to each sample, but certain samples carry more information than others. Our proposed similarity measure takes this notion into account and weights each sample by its estimated information content.

5.1.3 Chapter Specific Notation

This chapter studies similarity of measurement sequences. Navigation by indexing within data sets have to be carefully considered. For a quick orientation, see Figure 5.1. Each data sequence consists of one-dimensional measurement samples. The following notation is used:

- Single measurement at time instance t within k-th sequence is a scalar y_t^k.
- k-th sequence of measurements of length T_k is $\mathbf{Y}_{T_k}^k = \left(y_1^k, y_2^k, ..., y_{T_k}^k\right)$, in this chapter, it is a $1 \times T_k$ row vector.
- Set of all segment candidates is denoted as $\mathcal{Y} = \{\mathbf{Y}_{T_1}^1, \mathbf{Y}_{T_2}^2, ...\mathbf{Y}_{T_k}^k, ..., \mathbf{Y}_{T_K}^K\}$.

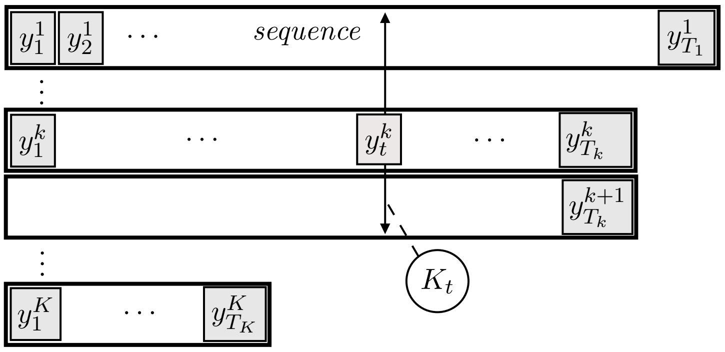


Figure 5.1: Sequence data set illustration

The measured sequence is considered to be a stochastic process. In this chapter, superscripting by a variable always means indexing of segment candidate unless explicitly stated otherwise.

Note that in this chapter, we deal with only scalar measurements y_t^k, thus the sequence of measurement $\mathbf{Y}_{T_k}^k$ is a flat matrix. We keep this notation to remain more or less consistent between chapters.

5.2 Clustering for Data Set Preparation

5.2.1 Problem Definition

The problem is decomposed into following three steps.

- Generate candidates for segments,
- Construct a suitable distance function,
- Classify similar candidates into the same class of operations.

A typical time series is depicted in Figure 5.2. A notable plateau divides the two candidates for segmented operations. There are also artifacts in data which do not present any robotic operation. For example, transition peak in power caused by robot waking up from power-saving mode. Other non-functional segments of activity in power consumption are caused by homing the robot before its shutdown. These artifacts have to be filtered out as noise in data.

Another issue with the segment candidates is that candidates do not have the same length. Not even within the same class of operation and they still must be clustered together. This issue

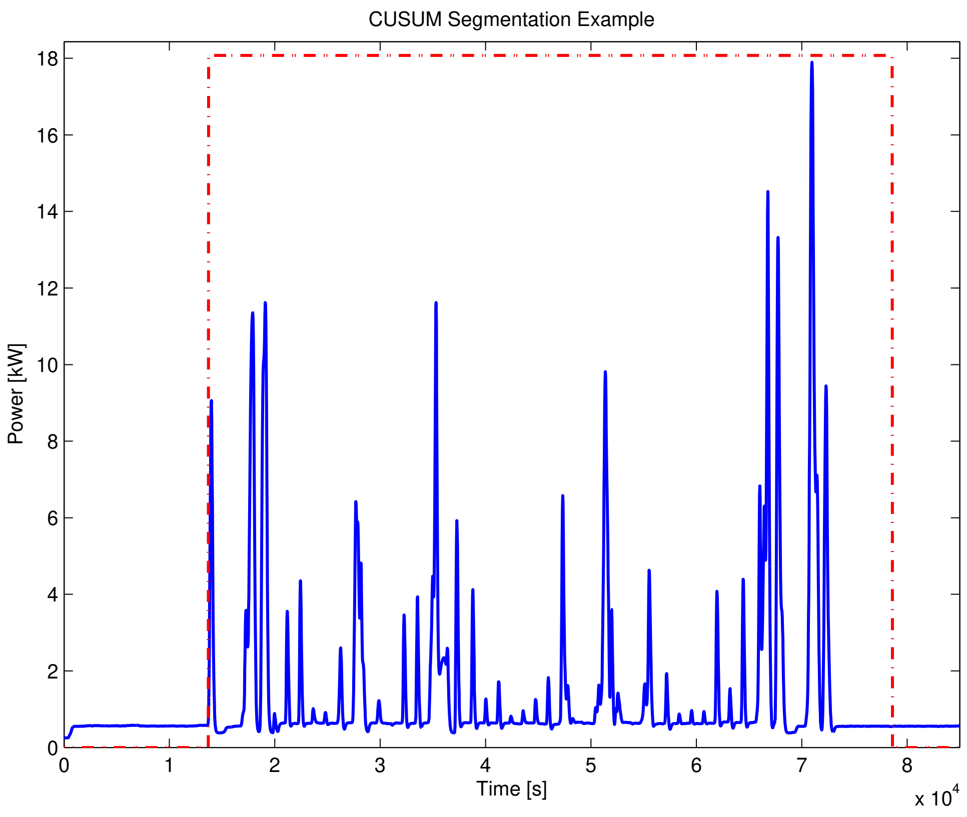


Figure 5.2: Data example and its CUSUM segmentation.

puts demands on the similarity measure computed for any pair of candidates. Some robotic operations can even be paused in the middle for an extended time period before its execution is resumed. Such fragments are difficult to cluster.

5.2.2 Generation of Candidate Segments

The initial segmentation is based on the observed property of power data which indicates that robotic operations are separated from each other by considerably long plateaus of low power consumption. The measured signal is rather steady during a plateau. A simple thresholding of data yields a lot of faulty separation points within valid operations. A near-zero value detection of noise-filtered differentiation of data fails due to the presence of short plateaus within valid operations. Such plateaus happen for example when the robot is spot welding.

The CUSUM algorithm for a detection of a change of a fundamental mode of the measured signal is implemented. The CUSUM algorithm operates over the ratio of the likelihoods of the hypothesis that *robot is operating* over the likelihood of the hypothesis *robot rests.* Those likelihoods are evaluated based on the difference signal and are expected to be normally distributed. The prior distribution was estimated based on manually chosen short sequences of both mode types mentioned above.

Supposing we have the likelihood of each sample in the sequence for both hypothesis as $l(\theta_0|y_t), l(\theta_1|y_t)$, where θ_0 represents the hypothesis that y_t belongs to a plateau (i.e., *robot rests*) and θ_1 represents the hypothesis that y_t belongs to operation sequence (i.e., *robot is operating*). The observed statistic S_t for CUSUM takes form of

$$s_t = \log \frac{l(\theta_0|y_t)}{l(\theta_1|y_t)}, \tag{5.1}$$

$$S_t = S_{t-1} + s_t. \tag{5.2}$$

The adaptive threshold for statistic S_t is introduced as

$$\begin{aligned} min_t &= \min_{i \in \mathcal{T}} S_i, && (5.3) \\ max_t &= \max_{i \in \mathcal{T}} S_i, && (5.4) \end{aligned}$$

where $\mathcal{T}$ represents the set of time indices since the last mode-change detection.

The decision that the mode of signal has changed is based on the following expression.

$$\begin{aligned} Rests &\Leftrightarrow S_k - min_t > \lambda_{Rests \to Operates}, && (5.5) \\ Operates &\Leftrightarrow S_k - max_t < -\lambda_{Operates \to Rests}, && (5.6) \end{aligned}$$

where $\lambda_{Rests \to Operates}$ is a tuning constant for change in the direction from the resting mode to the operating mode of the robot and the $\lambda_{Operates \to Rests}$ vice versa.

In Figure 5.2 is an example of CUSUM segmentation. By tuning the lambdas, the lag in the mode change detection can be tuned.

5.2.3 Distance of Candidate Segments

Both connectivity and density based clustering methods depend heavily on the distance definition used. The previous work in [10] with robotic power consumption type of data in showed that Pearson's coefficient is successful measure of similarity. In this work the Pearson's correlation coefficient is used as a basis of semimetric distance measure between segments. It is advantageous to express the computations in a form of vector-matrix operations as vectorization can significantly boost computational performance on modern computers.

Pearson's correlation coefficient for a pair of sequences $\mathbf{Y}^j_{T_j}$, $\mathbf{Y}^k_{T_k}$ of the same length $T = T_j = T_k$ is defined as

$$\begin{aligned} \rho(\mathbf{Y}^j_T, \mathbf{Y}^k_T) &= \frac{\sum_{t=1}^T (y^j_t - \mu^j)(y^k_t - \mu^k)}{\sqrt{\sum_{t=1}^T (y^j_t - \mu^j)^2 \sum_{t=1}^T (y^k_t - \mu^k)^2}} && (5.7) \\ &= \frac{\sum_{t=1}^T (y^j_t y^k_t) - T\mu^j \mu^k}{T\sigma^j \sigma_k}, && (5.8) \\ &= \frac{\mathbf{Y}^j_T {\mathbf{Y}^k_T}^\top - T\bar{y}^j \bar{y}^k}{T\sigma^j \sigma^k} && (5.9) \\ \sigma^k &= \sqrt{\sum_{t=1}^{T_k} \left(y^k_t\right)^2 - \left(\sum_{t=1}^{T_k} y^k_t\right)^2} && (5.10) \\ &= \sqrt{\mathbf{Y}^k_{T_k} {\mathbf{Y}^k_{T_k}}^\top - \left(\mathbf{Y}^k_{T_k} \mathbf{1}^k\right)^2} && (5.11) \end{aligned}$$

where μ^k is the mean of k-th candidate segment $\mathbf{Y}^k_{T_k}$ and σ^k is the standard deviation of k-th candidate segment $\mathbf{Y}^k_{T_k}$ and $\mathbf{1}$ is a $T^k \times 1$ column vector of ones.

The candidate sequences can differ in lengths and also in time lag with respect to each other. This issue can be addressed by extracting features from sequence such that the number of features is equal for any length of segment. Frequency analysis provides such feature generation but as discussed in [10] it does not yields satisfying similarity measure. Computing numerous statistical moments would also be almost independent of the length of the sequence but noise makes such statistic inconclusive.

In this work a distance is defined as a maximum correlation in sliding window normalized by the convolution of the information content of the overlapping regions of the compared candidate segments. Let $I(y_t^k)$ be an information content of sample y_t^k and let $\mathbf{I}(\mathbf{Y}_{T_k}^k)$ be a sequence of information contents of the sequence $\mathbf{Y}_{T_k}^k$ defined as

$$\begin{aligned} I(y_t^k) &= -\log p(y_t^k), & (5.12)\\ \mathbf{I}(\mathbf{Y}_{T_k}^k) &= \left[I(y_1^k), I(y_2^k), ..., I(y_{T_k}^k)\right]^\top, & (5.13)\\ \mathbf{I}^k &= \mathbf{I}(\mathbf{Y}_{T_k}^k), & (5.14)\end{aligned}$$

where $p(y_t^k)$ is the probability density function of the random variable Y and y_t^k is its realization at time t. To define the correlation vector, following functions need to be defined first.

$$\tilde{\mathbf{Y}}_{T_k}^k = \mathbf{Y}_{T_k}^k - \mu^k, \qquad (5.15)$$

where μ^k is the mean value of the sequence $\mathbf{Y}_{T_k}^k$ and the $\tilde{\mathbf{Y}}_{T_k}^k$ is the centered sequence. Using the discrete convolution $(*)$ and the swap of ordering (prime sign) following variables are defined as

$$\begin{aligned} \mathbf{q}^{jk} &= \tilde{\mathbf{Y}}_{T_j}^j * \tilde{\mathbf{Y}}_{T_k}^{k\,\prime}, & (5.16)\\ \boldsymbol{\sigma}^j &= \mathbf{1}^k * \left(\tilde{\mathbf{Y}}_{T_j}^j\right)^2, & (5.17)\\ \boldsymbol{\sigma}^k &= \mathbf{1}^j * \left(\tilde{\mathbf{Y}}_{T_k}^k\right)^2, & (5.18)\end{aligned}$$

where $\mathbf{1}^k$ represents $T_k \times 1$ vector of ones and $\left(\tilde{\mathbf{Y}}_{T_k}^k\right)^2$ means element-wise square of elements of the vector $\tilde{\mathbf{Y}}_{T_k}^k$.

The normalization vector $\mathbf{r}^{jk}$ is defined as

$$\mathbf{r}^{jk} = \frac{\left(\mathbf{1}^j * \mathbf{I}^k\right) + \left(\mathbf{1}^k * \mathbf{I}^j\right)}{\mathbf{1}^j\mathbf{I}^j + \mathbf{1}^k\mathbf{I}^k}. \qquad (5.19)$$

Finally using the element-wise product $\odot$ and the element-wise right-division $\oslash$, the correlation vector $\boldsymbol{\rho}(\mathbf{Y}_{T_j}^j, \mathbf{Y}_{T_k}^k)$ is defined.

$$\boldsymbol{\rho}(\mathbf{Y}_{T_j}^j, \mathbf{Y}_{T_k}^k) = \mathbf{r}^{jk} \odot \mathbf{q}^{jk} \oslash \sqrt{\boldsymbol{\sigma}^j \odot \boldsymbol{\sigma}^k}, \qquad (5.20)$$

where the square root is also calculated element-wise. All the vectors have exactly the same dimensions and none of the elements of the expression $\sqrt{\boldsymbol{\sigma}^j \odot \boldsymbol{\sigma}^k}$ is zero, so that all the element-wise operations are well-defined.

The elements of the resulting vector $\boldsymbol{\rho}(\mathbf{Y}_{T_j}^j, \mathbf{Y}_{T_k}^k)$ are in fact the correlation functions (5.7) of the overlapping regions as the sequences $\mathbf{Y}_{T_j}^j, \mathbf{Y}_{T_k}^k$ slide over each other weighted by the fraction of information contained in both overlapped regions over the total information contained in both entire sequences. In Figure 5.3 is depicted an example of a correlation vector of two sequences with its maximum at the shift of 394 between them.

The final distance is defined as

$$D\left(\mathbf{Y}_{T_j}^j, \mathbf{Y}_{T_k}^k\right) = \frac{1}{2}\left(1 - \max_{\rho} \boldsymbol{\rho}\left(\mathbf{Y}_{T_j}^j, \mathbf{Y}_{T_k}^k\right)\right). \qquad (5.21)$$

The $D\left(\mathbf{Y}_{T_j}^j, \mathbf{Y}_{T_k}^k\right)$ denotes the distance between two candidate segments $\mathbf{Y}_{T_j}^j, \mathbf{Y}_{T_k}^k$. The distance defined as (5.21) does not obey the triangle inequality but it is always greater or equal to zero, it is a symmetric measure and it is zero only when the candidate segments $\mathbf{Y}_{T_j}^j, \mathbf{Y}_{T_k}^k$ are equal. For these reasons this distance is only a semi-metric. Note that we agree on that the segments are similar when they have a constant offset from each other. The distance can be zero also in the case when one segment is just an affine transformation of the other segment. A very slow drift was observed in the data and it is desirable feature of the distance function that it is insensitive to the linearly expressed drift. There are also special cases some segment is just an affine transformation of another one and yet they do not belong into the same class. But such events were not observed in the data.

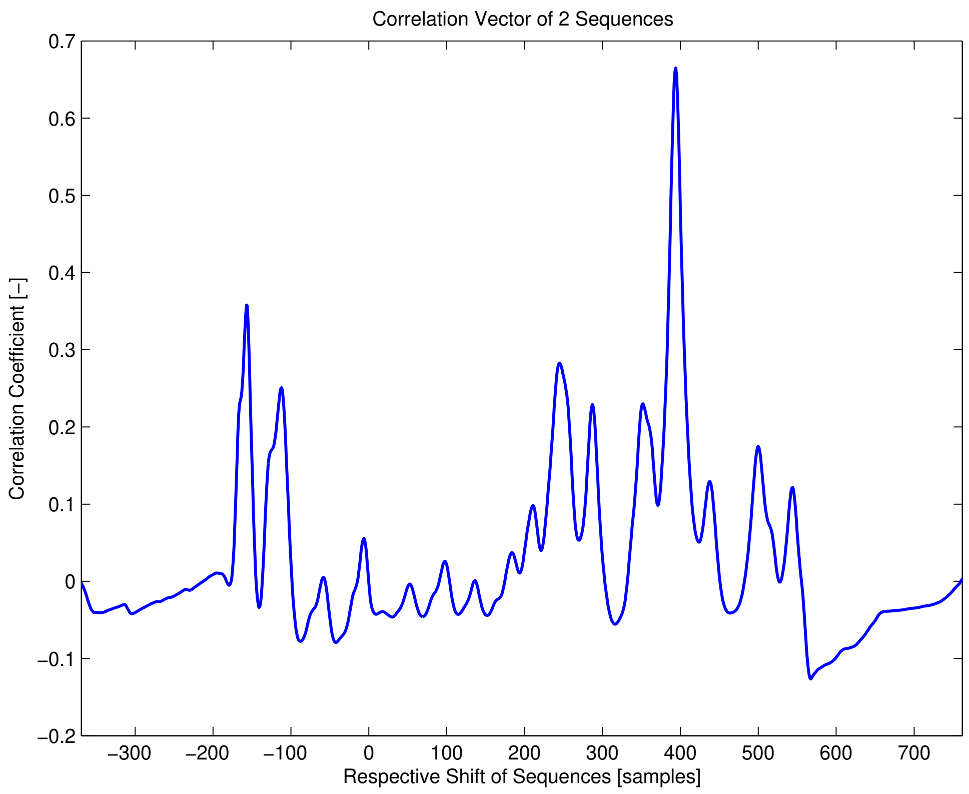


Figure 5.3: Correlation vector of two sequences.

5.2.4 Density Based Clustering

By terminology of clustering theory, the candidate segment is denoted as a point. An agglomerative linkage and also density-based clustering methods were investigated for an unsupervised clustering of the segment candidates. The clustering methods were selected so that a minimum number of tuning parameters were necessary. First, connectivity-based hierarchical clustering was tested, a single-linkage clustering in particular. Its advantage is that it can find even very irregularly shaped clusters whereas other popular clustering methods such as Gaussian distribution based clustering struggle with data distributed other than within a range of some hypersphere.

Next, more complicated clustering methods were investigated. First, the popular DBSCAN method showed successful clustering after carefully tuning the control parameters. DBSCAN behavior is determined by two parameters,

- ε - neighborhood range,
- $minPts$ - minimum number of neighbors in ε range.

The parameter ε determines what is the maximal distance between the closest neighbors so that they are considered to be connected. The parameter $minPts$ determines minimum connected neighbors that some point needs to be considered a member of a cluster. This way, points which do not fit those rules are classified as noise, which is the desired behavior as discussed earlier. To overcome issues of uncertainty in the choice of parameter ε a generalized DBSCAN method called OPTICS (see [78] [79]) was utilized. OPTICS does not need initialization of ε. It is run for the maximal sensible setting of ε and the algorithm shows in its reachability plot how the clustering is affected by the choice of ε. This way, ε is determined flexibly after the computationally expensive phase of the clustering. Let N be the number of points. The theoretic time complexity of the OPTICS algorithm is $\mathcal{O}(N^2)$ due to the computation of the nearest neighbors of each point in the dataset. For completely ordered spaces, the complexity can be reduced to $\mathcal{O}(N \log N)$.

The pseudocode for the OPTICS algorithm is depicted in Algorithm 1. Two distances are used in the OPTICS.

- Core distance
- Reachability distance

Core distance of the point is the distance to its *minPts*-th closest neighbor in ε range. If there is not that many neighbors in range, the core distance is undefined.

The reachability distance of a pair of points can be defined only when there is defined core distance of the first point. If there is enough neighbors in ε range, core distance is defined and the reachability distance is $\max\{CoreDist(point_1), Dist(point_1, point_2)\}$.

Algorithm 1: OPTICS clustering algorithm.

```
Data: DB, $\varepsilon$, minPts
Result: Reachability distance order queue
foreach point in DB do
    point.reachDist ← $\emptyset$;
foreach unprocessed point in DB do
    N ← GetNeighbors(point, $\varepsilon$);
    mark point as processed;
    append point to output ordered queue;
    if CoreDistance(point, $\varepsilon$, minPts) $\neq \emptyset$ then
        Seeds.Init();
        Update(N, point, Seeds, $\varepsilon$, minPts);
        foreach next point' in Seeds do
            N' ← GetNeighbors(point', $\varepsilon$);
            mark point' as processed;
            append point' to output ordered queue;
            if CoreDistance(point', $\varepsilon$, minPts) $\neq \emptyset$ then
                Update(N', point', Seeds, $\varepsilon$, minPts);
```

The update function's pseudocode is showed in Algorithm 2. The ordered set Seeds holds the list of unprocessed points ordered by their reachability distance to the point which included them as neighbors. When a new reachability distance is assigned to the point, it must be put into correct place in Seeds i. e. Seeds must be sorted according to the reachability distances. When taking the next point from Seeds, next is the point with the smallest reachability distance available. After a point is taken as next point, it is excluded from Seeds. The OPTICS run ends by producing an ordered queue of distance labeled points. The ordering is visualized by reachability plot as in Figure 5.4. Each bar in Figure 5.4 corresponds to a point, and the height of the bar shows the largest distance at which the point will still be included in some cluster. Each point is traced back from its representation within the reachability plot.

The valleys in the reachability plot correspond to the clusters. The peaks divide the data points into separate clusters. The reader can see that the clustering depends on the choice of the threshold for cutting the reachability plot. The threshold represents the ε choice which had to be done before the clustering in the case of DBSCAN. For OPTICS supervisor can quickly iterate through many different settings of ε at negligible additional computational cost. One can see that as the threshold is lowered the number of clusters will rise and vice versa. After

Algorithm 2: Update method of the OPTICS.

```
Input: N, point, Seeds, $\varepsilon$, minPts
coreDist ← CoreDistance(point, $\varepsilon$, minPts);
foreach unprocessed neighbor in N do
    newReachDist ← Max(coreDist, Distance(point, neighbor));
    if neighbor.reachDist = $\emptyset$ then
        neighbor.reachDist ← newReachDist;
        Seeds.Insert(neighbor);
    else
        if newReachDist < neighbor.reachDist then
            neighbor.reachDist ← newReachDist;
            Seeds.Sort(neighbor);
```

setting the threshold, the points with reachability distance above the threshold are discarded as noise except for the most-right one in each peak. Also, the clusters counting less than *minPts* are filtered out as noise.

The peaks in Figure 5.4 are very distinctive, which is caused by a presence of points on the edge of clusters. It means that the given candidate segments contain outlying segments which are not similar to more than $minPts - 1$ other candidate segments. That is caused by imperfect preliminary segmentation of the data into candidate segments. It can also be caused by the occurrence of events in the robotic cell that happens rarely and are captured in the data at most $minPts - 1$ times.

5.3 Experiments

In Figure 5.2, there is an example of the performance of the candidate segment generation algorithm. The segments framed by the 1-valued red-dashed criterion function are considered to be data generated by an operation of the robot. The mode-detection CUSUM algorithm is able to segment measured data with sufficient lag after the robot mode switches into the resting mode. As there are two tuning parameters λ, the algorithm is shown to be more sensitive to the transition from the resting mode into the operating mode. This was necessary due to the fact that the segmentation evaluated the plateaus during the spot-welding phases as resting mode of the robot. This can be hardly overcome by such simple signal filtering and in this work an effort was focused on reducing the impact of imperfections of segmentation phase. Measurement data of one day of production were processed by the whole chain of the preprocessing tools described above. The data contain 459 operation executions.

In Figure 5.5, there is an example of resulting clustering. Only three representative examples from each cluster are plotted. It can be seen that shifts and scaling in the time domain of the operation candidates cause growth in overall dissimilarity of the candidate segments. It leads to the creation of multiple classes which can be identified as equal by eye. This issue is caused by two main reasons. One is that the distance function to marks the unshifted segments as closer than the shifted ones. The second and lesser reason is simple thresholding of the reachability plot. More complicated methods for the valley detection could improve the clustering.

5.4 Conclusion

In this chapter, we have shown how clustering can be used to prepare training data sets. We examined approaches requiring minimum manual tuning from a human user. The essential contribution is the similarity measure that can compare a pair of unequal-length vectors and

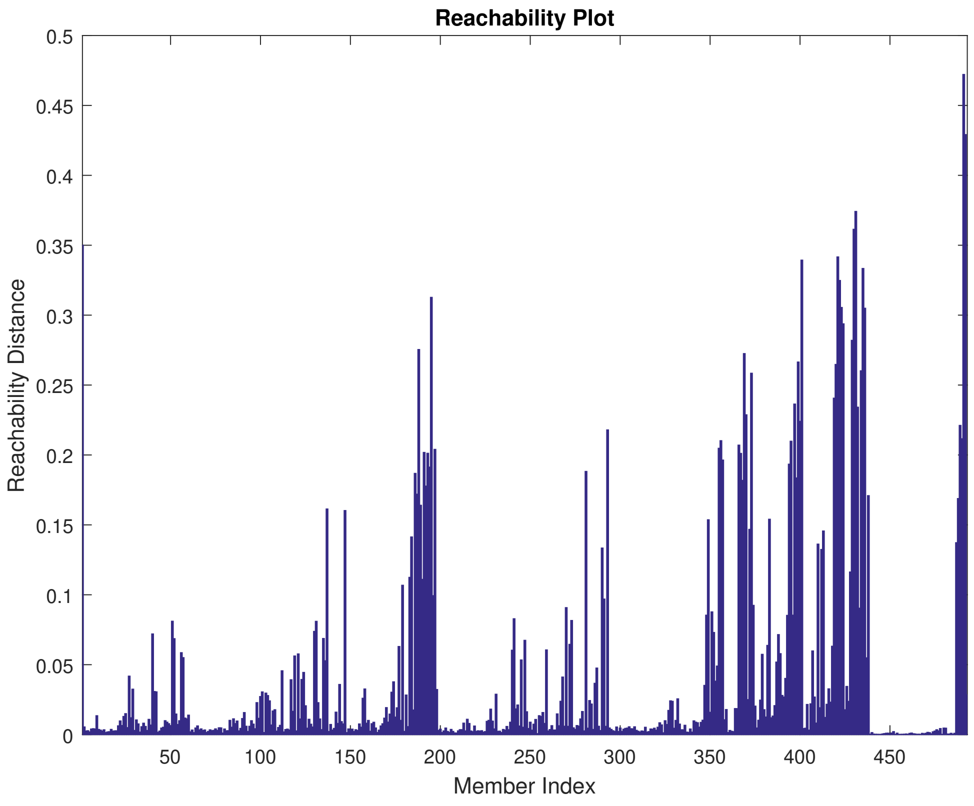


Figure 5.4: Reachability plot of OPTICS clustering with $minPts = 6$.

return a percentage-like result, which is easy to interpret. The OPTICS algorithm proved to be a valuable tool in tuning the algorithm parameters. It also brings a better insight into the processed data. Additionally, we introduced a signal segmentation strategy based on its information content.

Both the segmentation step and the similarity measure are modular. They can be replaced by different methods, but regarding the clustering algorithms, OPTICS showed the most suitable set of features compared to other density-based clustering techniques. The segmentation performance was negatively affected by behavior-agnostic signal processing. How to improve the segmentation is the topic we focus on in the next chapter.

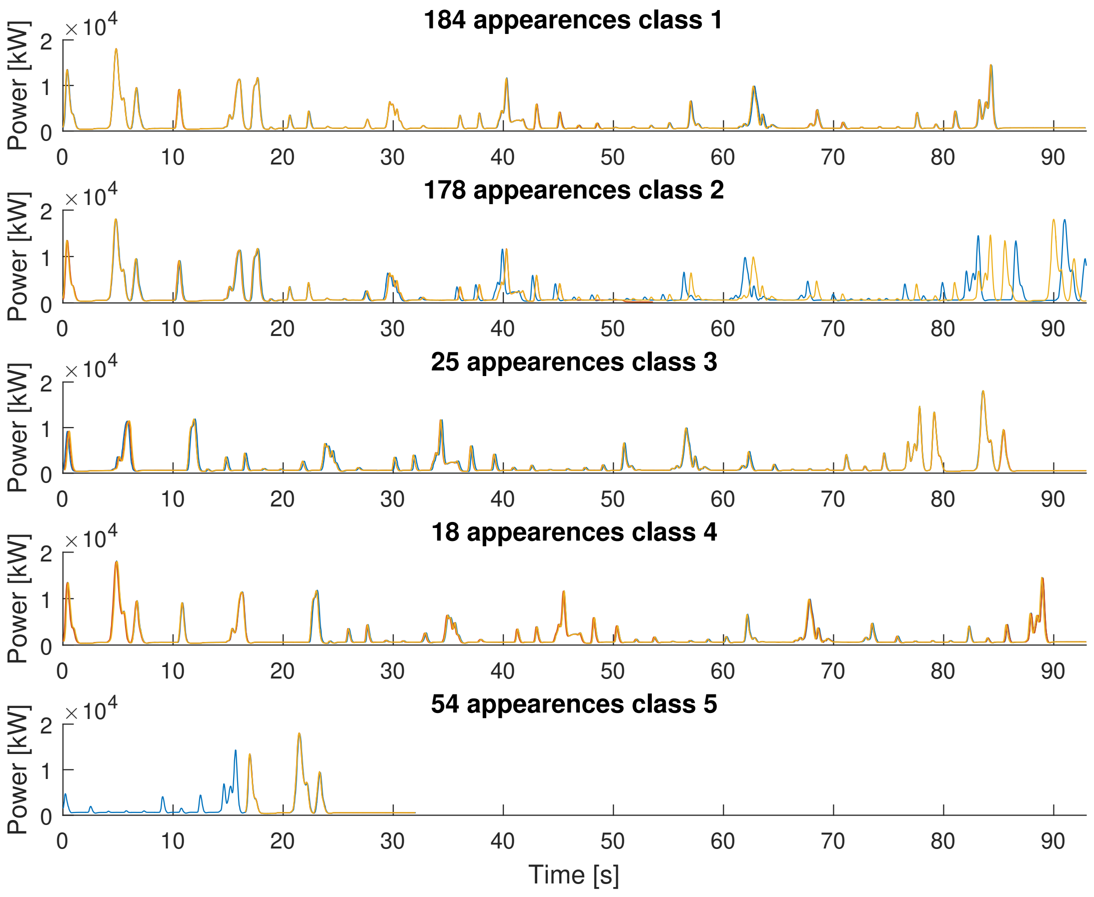


Figure 5.5: Example of clustered segments with allowed distance 0.25.

Chapter 6

Pattern Modeling

The pattern model described in Chapter 3 relies on the correlation between the manually discovered pattern and the sliding window going through the time series. Such an approach can be viewed as a KNN classification task, where each window is viewed as a point in a high-dimensional space. However, one of the objectives of this dissertation is to provide classifications together with an estimate of the prediction uncertainty. Even though there exist methods to express the confidence (complement of uncertainty) in predictions of the KNN classifier, stochastic models provide a well-established implicit toolset for uncertainty, making them better suited for the main goals of this dissertation work. This chapter investigates a particular stochastic model – the Hidden Markov model. We present an idea to include the exogenous control input into the hidden state of the HMM. To our best knowledge, such an approach is not widely used in the industry despite its modeling strength. This chapter is based on our original work in [3].

6.1 Introduction

This chapter deals with the identification of robotic operations for industrial robots that are used in robotic manufacturing cells such as welding or assembling ones. In such cases the robotic operations are planned in advance and executed as robotic programs that run in the robot controller and make the robot move and perform various tasks such as manipulation, welding, glueing etc. The planning procedure specifies the expected duration of the operations as well as their dependencies. The actual shape of the trajectories belonging to the individual operations and their duration are specified later during offline or online robot programming. If offline programs are done in a simulation tool such as Robcad or Tecnomatix Process Simulate the entire cell may be simulated in a time-based or event-based manner. The latter one allows modelling and simulating actual robot dependencies too but a more important fact is that the simulation allows measuring the actual duration of the robot operations, the production cycle of the cell and other qualities.

The motivation is the requirement to recognise the individual robotic operations based on various measurements during the actual operation of the robots. There may be various reasons for this, we name just a few: (a) to watch the individual operations and compare them to the expected behaviour, (b) to watch the current state of the production cell and check if it operates correctly, or (c) to predict the behaviour of the production cell in terms of e.g. pauses that are not planned in advance and that may be caused by a coincidence of various (unwanted) events. All the above mentioned situations relate to the diagnostics of the robots or of the whole production cell. The measurements that provide values for the identification may be of different nature. It is useful if the dynamic behaviour of the robots is taken into account but it is only rarely the case. It turns out that the most achievable way how to measure the robot behaviour is to get its power consumption. Such a measurement can be achieved by adding specialised power measurement cards that measure the voltage and current consumed by the robot at its power inlet. Its disadvantage is that it does not take account of the internal robot dynamics since it measures the consumed power of the whole robot but not of its individual axes. One possible way of overcoming this disadvantage is to obtain internal operational indicators from the robot controller such as axis position, acceleration etc. However this is not always possible. Another possible way is to have a model of the robot that represents its structure and that is able to compute or to estimate the robot state and the inputs that lead to the state.

We propose a model of a robot based on time-varying hidden Markov chains that are used to classify the output sequences into classes of input sequences that have generated the outputs. The output sequences are obtained as sampled measurements of the robot energy consumption and the input sequences correspond to the robot trajectories that represent the robot operations. The robot is described by a stochastic state space model and the input sequence is assumed to be independent of the robot state in terms of its state space model. The input sequence is modelled as a discrete time-varying Markov model.

Unlike other approaches (see e.g. [10] or [80]) it allows recognising the input sequences even with a system that is not observable (due to the hidden inputs). If the model of the system is known, more insight into its internal behaviour can be obtained based on the Markov modeling. This may be useful not only for the identification of the robot operations but also for diagnosis of changes of the robot behavior with respect to the expectation.

6.1.1 Related Work

Previous work [10] dealing with the identification of robotic operations provided an algorithm suitable for online and thus fast recognition of the patterns that correspond to the individual robotic operations. It was based on the feature extraction whereas the features are the peaks in the measured power consumption data. The results of the algorithm have been strongly dependent on the correct selection of the master patterns which had to be done manually by adding marks into the input data that signal the borders between consecutive operations. Additionally, if the number of peaks changes during the robot operation or the threshold does not suit any more because of various possible offsets then the hit ratio of the pattern-recognition algorithm decreases.

This work focuses on estimating the inputs and system states based on the measurement of the outputs. For observable systems there have been several works such as [80], which extends the work of [81], to estimate the unknown inputs and the system state using an adapted Kalman filter.

The hidden Markov models are widely used for pattern recognition in the fields like speech and image processing. There are also numerous publications devoted to using Markov models for mobile robot orientation and navigation, human-robot interaction, etc. Regarding industrial manipulators there have been approaches using Markov models to teach a robot how to mimic human behaviour such as in [82], to design a feedback controller such as in [83] etc.

The specifics of the work presented in this paper lie in the fact that the robot is modeled as a system that is not observable. This corresponds to real-life cases occurring in robotic manufacturing facilities. Still it is an important requirement to teach the underlying Markov models properly. This process is typically application-specific while using general methods such as maximum likelihood optimisation. In theoretical or in-laboratory situations it may be possible to perform various experiments to obtain the robot dynamic parameters by dynamic model identification. There have been numerous publications in this field that take into account the dynamic model of the whole robot such as in [84], [85], [86] or [87]. In existing robotic cells it may be infeasible to perform various experiments with the robots simply due to spacial constraints not allowing to perform arbitrary robotic movements. In such cases an useful method is to construct the dynamic model analytically by physical modeling of the robot as described in [9] or [88]. It is also possible to use a simulation environment that supports Realistic Robotic Simulations (see [89]). For the purpose of this paper we utilise a simplified model of one robotic axis and perform in-lab experiments with a 6-DOF robot to demonstrate the usability of our approach.

6.1.2 Chapter-Specific Notation

The following specific notation is used in this chapter.

- Lower index t is used as a discrete-time sample index.
- $p_m(\mathbf{x}) = p(\mathbf{x}|m)$ is the conditional probability of $\mathbf{x}$ given m.
- $\mathbf{X}_T = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T)$ is a sequence of $\mathcal{X}^a$-valued random vectors, where a is the dimension of the vector $\mathbf{x}_t$ and $\mathcal{X}$ is the set of possible values for each element of vector $\mathbf{x}_t$.
- $p(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T) = p(\mathbf{X}_T)$ is the joint probability of sequence of vectors $\mathbf{x}_t$.

In this chapter, probabilities $p(.)$ are pmf of discrete random vectors. It is a function assigning a scalar value to a point from its domain. For example $p(\mathbf{x}, \mathbf{y}), \mathbf{x} \in \mathcal{X}^a, \mathbf{y} \in \mathcal{Y}^b$ is a joint-probability function $p(\mathbf{x}, \mathbf{y}) : \mathcal{X}^a \times \mathcal{Y}^b \to [0, 1] \subset \mathbb{R}$.

6.2 Pattern-Classification Model

6.2.1 Problem Definition

Let us suppose we have an industrial robotic manipulator with a predefined set of robotic operations and we measure the power consumption of the robot. These operations are in general the desired trajectories of the robot which are meant in terms of the robotic effector. Each trajectory is executed by the robot controller which generates necessary torques for each of the robot axes according to its internal trajectory planning and interpolation algorithms. These torques correspond to the robot internal states that are not known because the power consumption of the whole robot and not of the individual axes is measured.

Suppose that the dynamic system of the robot can be described using a discrete state-space model

$$\begin{aligned} \mathbf{x}_{t+1} &= \mathbf{f}\left(\mathbf{x}_t, \mathbf{u}_t\right) + \mathbf{v}_t, && (6.1)\\ \mathbf{y}_t &= \mathbf{g}(\mathbf{x}_t, \mathbf{u}_t) + \mathbf{e}_t, && (6.2) \end{aligned}$$

where $\mathbf{f}$ is a vector-valued state-transition function, $\mathbf{x}$ is a state variable representing the angle and angular velocity for each axis, $\mathbf{u}$ is an input variable representing the torque applied on an axis. $\mathbf{v}$ and $\mathbf{e}$ are Gaussian white noise variables and $\mathbf{y}$ represents the measured power consumption. Particularly $\mathbf{v}$ and $\mathbf{e}$ are a process noise and a measurement noise respectively. The stochastic description of the system comes from the transformation of the random variables.

$$\begin{aligned} p\left(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t\right) &= p_v\left(\mathbf{x}_{t+1} - \mathbf{f}\left(\mathbf{x}_t, \mathbf{u}_t\right)\right), && (6.3)\\ p\left(\mathbf{y}_t|\mathbf{x}_t, \mathbf{u}_t\right) &= p_e\left(\mathbf{y}_t - \mathbf{g}\left(\mathbf{x}_t, \mathbf{u}_t\right)\right). && (6.4) \end{aligned}$$

The evolution of the model from (6.3) and (6.4) can be described as a Hidden Markov Model with structure depicted in Figure 6.1, where the hidden input sample $\mathbf{u}_t$ together with the hidden state-space variable $\mathbf{x}_t$ form a hidden superstate $\mathbf{s}_t$.

This superstate $\mathbf{s}_t$ produces an output sample $\mathbf{y}_t$ which is observed. The structure depicted here emphasizes that $\mathbf{u}_t$ is independent of $\mathbf{x}_t$. Due to that the Markov model of input sequences $\mathbf{U}_T$ and the transition model of states $\mathbf{x}_t$ can be constructed separately. Markov model is described in terms of superstate $\mathbf{s}_t$ by transition probabilities $p_m(\mathbf{s}_t|\mathbf{s}_{t-1})$ defined as

$$\begin{gathered} p_m(\mathbf{s}_t|\mathbf{s}_{t-1}) = p_m(\mathbf{x}_t, \mathbf{u}_t|\mathbf{x}_{t-1}, \mathbf{u}_{t-1}), \qquad (6.5)\\ \mathbf{u}_t \in \mathcal{U}^q,\ \mathbf{x}_t \in \mathcal{X}^a,\ \mathbf{s}_t \in \mathcal{K} = \mathcal{X}^a \times \mathcal{U}^q, m \in \mathcal{M}, \end{gathered}$$

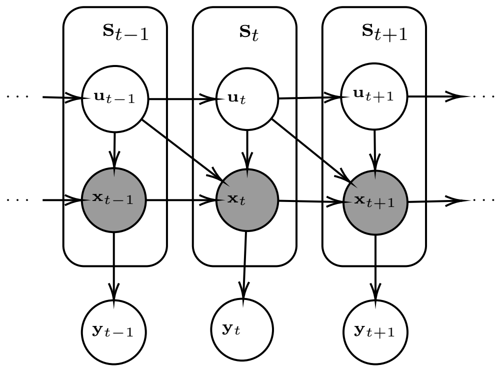


Figure 6.1: Proposed HMM structure in the form of dependency Bayes net

where q, a are the numbers of inputs and state variables of the dynamic system, respectively. Sets $\mathcal{U}$, $\mathcal{X}$ and $\mathcal{K}$ represent the set of possible input values, the set of possible state values and the set of all combinations of input and state values, respectively. $\mathcal{M}$ is the set of all distinctive Markov models of the input sequence classes.

Assuming that the input samples are independent of the states of the system and that the state evolution depends only on the previous sample $\mathbf{x}_{t-1}$ and $\mathbf{u}_{t-1}$, then from conditional input independence described in Figure 6.1 and from (6.5) using the chain rule we derive

$$\begin{aligned} p_m(\mathbf{x}_t, \mathbf{u}_t | \mathbf{x}_{t-1}, \mathbf{u}_{t-1}) &= p_m(\mathbf{u}_t | \mathbf{x}_t, \mathbf{x}_{t-1}, \mathbf{u}_{t-1}) p(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{u}_{t-1}) \\ &= p_m(\mathbf{u}_t | \mathbf{u}_{t-1}) p(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{u}_{t-1}). \end{aligned}$$

Here term $p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_{t-1})$ represents the stochastic model of the dynamic system. Notice that this probability is not dependent on the (Markov) model of any input sequence class $m \in \mathcal{M}$.

Term $p_m(\mathbf{u}_t|\mathbf{u}_{t-1})$ describes the Markov model of a m-th input sequence class corresponding to a m-th trajectory. For this assignment set $\mathcal{M}$ of the models is considered to be given.

The probability of observing sample $\mathbf{y}_t$ of observed feature sequence $\mathbf{Y}_T$ is

$$p(\mathbf{y}_t|\mathbf{s}_t) = p(\mathbf{y}_t|\mathbf{x}_t, \mathbf{u}_t). \tag{6.6}$$

The probability of observing a sequence of outputs $\mathbf{Y}_T$ for a given input model $m \in \mathcal{M}$ is

$$p_m(\mathbf{Y}_T) = \sum_{\{\mathbf{s}_1 \in \mathcal{K}, \mathbf{s}_2 \in \mathcal{K}, \ldots, \mathbf{s}_T \in \mathcal{K}\}} p_m(\mathbf{Y}_T, \mathbf{S}_T), \tag{6.7}$$

$$p_m(\mathbf{Y}_T, \mathbf{S}_T) = \prod_{t=1}^{T} p_m(\mathbf{s}_t|\mathbf{s}_{t-1}) p(\mathbf{y}_t|\mathbf{s}_t), \tag{6.8}$$

$$p_m(\mathbf{s}_1|\mathbf{s}_0) \triangleq p_m(\mathbf{s}_1).$$

The observed sequence $\mathbf{Y}_T$ is then decided to be generated by the input sequence from class m with the highest probability $p_m(\mathbf{Y}_T)$.

6.2.2 Parameters Inference

To classify a sequence, we need to calculate marginal probability $p_m(\mathbf{Y}_T)$ from equation (6.7) for each model $m \in \mathcal{M}$ and select $m^* \in \mathcal{M}$ such that $p_m(\mathbf{Y}_T)$ is maximal:

$$m^* = \arg\max_{m \in \mathcal{M}} p_m(\mathbf{Y}_T). \tag{6.9}$$

The following definition and substitution are introduced

$$p_m(\mathbf{s}_1|\mathbf{s}_0)p(\mathbf{y}_1|\mathbf{s}_1) \triangleq p_m(\mathbf{s}_1)p(\mathbf{y}_1|\mathbf{s}_1), \tag{6.10}$$

$$h_m(\mathbf{y}_t, \mathbf{s}_t, \mathbf{s}_{t-1}) \triangleq p_m(\mathbf{s}_t|\mathbf{s}_{t-1})p(\mathbf{y}_t|\mathbf{s}_t). \tag{6.11}$$

Substituting (6.8) into (6.7) and then using (6.11) the resulting likelihood can be expressed as

$$p_m(\mathbf{Y}_T) = \sum_{\mathbf{s}_T\in\mathcal{K}} \cdots \sum_{\mathbf{s}_1\in\mathcal{K}} \prod_{t=1}^{T} p_m(\mathbf{s}_t|\mathbf{s}_{t-1})p(\mathbf{y}_t|\mathbf{s}_t) = \tag{6.12}$$

$$\sum_{\mathbf{s}_T\in\mathcal{K}} \sum_{\mathbf{s}_{T-1}\in\mathcal{K}} \cdots \sum_{\mathbf{s}_1\in\mathcal{K}} \prod_{t=1}^{T} h_m(\mathbf{y}_t, \mathbf{s}_t, \mathbf{s}_{t-1}). \tag{6.13}$$

Expression (6.13) can be rearranged by factoring out to the front of the sums as shown in equation (6.14).

$$\begin{gathered}
\sum_{\mathbf{s}_T\in\mathcal{K}} \sum_{\mathbf{s}_{T-1}\in\mathcal{K}} \cdots \sum_{\mathbf{s}_1\in\mathcal{K}} \prod_{t=1}^{T} h_m(\mathbf{y}_t, \mathbf{s}_t, \mathbf{s}_{t-1}) = \\
\sum_{\mathbf{s}_T\in\mathcal{K}} \sum_{\mathbf{s}_{T-1}\in\mathcal{K}} h_m(\mathbf{y}_T, \mathbf{s}_T, \mathbf{s}_{T-1}) \cdots \sum_{\mathbf{s}_1\in\mathcal{K}} \prod_{t=1}^{T-1} h_m(\mathbf{y}_t, \mathbf{s}_t, \mathbf{s}_{t-1}) = \\
\sum_{\mathbf{s}_T\in\mathcal{K}} \sum_{\mathbf{s}_{T-1}\in\mathcal{K}} h_m(\mathbf{y}_T, \mathbf{s}_T, \mathbf{s}_{T-1}) \cdots \sum_{\mathbf{s}_2\in\mathcal{K}} h_m(\mathbf{y}_2, \mathbf{s}_2, \mathbf{s}_1) \sum_{\mathbf{s}_1\in\mathcal{K}} h_m(\mathbf{y}_1, \mathbf{s}_1, \mathbf{s}_0)
\end{gathered} \tag{6.14}$$

In this factorisation it can be seen that the summation can be performed in parallel for each of the models. More importantly the summation can also be done recurrently. A statistics $\varphi(\mathbf{s}_t)$ is introduced

$$\varphi(\mathbf{s}_t) = \sum_{\mathbf{s}_{t-1}\in\mathcal{K}} \varphi(\mathbf{s}_{t-1})p_m(\mathbf{s}_t|\mathbf{s}_{t-1})p(\mathbf{y}_t|\mathbf{s}_t) \tag{6.15}$$

$$= \sum_{\mathbf{s}_{t-1}\in\mathcal{K}} \varphi(\mathbf{s}_{t-1})h_m(\mathbf{y}_t, \mathbf{s}_t, \mathbf{s}_{t-1}), \tag{6.16}$$

$$\varphi(\mathbf{s}_1) = p_m(\mathbf{s}_1|\mathbf{s}_0)p(\mathbf{y}_1|\mathbf{s}_1) = p_m(\mathbf{s}_1)p(\mathbf{y}_1|\mathbf{s}_1). \tag{6.17}$$

Using this consecutive summing, the likelihood is evaluated as

$$p_m(\mathbf{Y}_T) = \sum_{\mathbf{s}_T\in\mathcal{K}} \varphi(\mathbf{s}_T). \tag{6.18}$$

Let us break the superstate $\mathbf{s}_t$ into two pieces according to expressions (6.5)-(6.6). Note that from the nature of the task there will always be one particular observed output sequence examined at a time so there is no need to evaluate probabilities $p(\mathbf{y}_t|\mathbf{s}_t)$ for all possible values of $\mathbf{y}_t$ as it is needed for state variable $\mathbf{s}_t$. Instead only the probability of a particular realisation of $\mathbf{y}_t$ for each possible value of $\mathbf{s}_t$ will be needed. This fact is used to further simplify the calculations and will be emphasised by using $\mathbf{y}_t^\dagger$. Statistic φ can be understood as a table assigning some probability to each possible combination of variable $\mathbf{x}_t \in \mathcal{X}^a$ and $\mathbf{u}_t \in \mathcal{U}^q$ while value $\mathbf{y}_t^\dagger$ is fixed. Therefore this simplification in notation will be used

$$\varphi(\mathbf{x}_t, \mathbf{u}_t, \mathbf{y}_t^\dagger) \triangleq \varphi(\mathbf{x}_t, \mathbf{u}_t). \tag{6.19}$$

Expression (6.20) shows how $\varphi(\mathbf{x}_t, \mathbf{u}_t)$ is evaluated recursively in an efficient way.

$$\varphi(\mathbf{x}_t, \mathbf{u}_t) = p(\mathbf{y}_t^\dagger|\mathbf{x}_t, \mathbf{u}_t) \sum_{\mathbf{x}_{t-1}\in\mathcal{X}^a, \mathbf{u}_{t-1}\in\mathcal{U}^q} \varphi(\mathbf{x}_{t-1}, \mathbf{u}_{t-1})p_m(\mathbf{u}_t|\mathbf{u}_{t-1})p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_{t-1}) \tag{6.20}$$

The probabilistic meaning of intermediate result $\varphi(\mathbf{x}_t, \mathbf{u}_t)$ is

$$\varphi(\mathbf{x}_t, \mathbf{u}_t) = p(\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_t, \mathbf{x}_t, \mathbf{u}_t).$$

The likelihood evaluation can be done sequentially (online) as the samples from the observed sequence come. This likelihood can be used even for an intermediate classification with an increasing level of confidence as more samples are processed.

Assume that the observed output sequence is surely generated by one of the known input sequence classes. Supposing that each class of input sequences has the same prior probability $p(m)$, then the probability of the class for the given sequence can be expressed as

$$p(m|\mathbf{Y}_T) = \frac{p(\mathbf{Y}_T|m)p(m)}{\sum\limits_{m \in \mathcal{M}} p(\mathbf{Y}_T|m)p(m)} \propto p_m(\mathbf{Y}_T).$$

For the proof of concept, a simplified example of a dynamic system is measured. The example system, which represents a single axis of a robotic manipulator, is depicted in Figure 6.2. For the purpose of system modelling we initially used measurements of axial speed, position and driving torque of the robot's axis together with a power consumption. After having modelled the system, we restricted ourselves to measuring only the power consumption. The system is approached as non-linear. It can generally be described as a driven pendulum with friction which has 2 dynamic states (the angular speed and position) and 1 input (driving torque). The output of this system is then a quadratic function of torque driving the axis.

Here we present an experiment with 4 different robotic trajectories, each of them corresponding to its generating input sequence class. Each trajectory consists of the same number of points with the same ordering. The only difference is the position of the points which varies in a range of 1 cm. Each trajectory was measured 200 times creating data corpus of 800 sequences. The corpus was divided into a training set (75% of the corpus) and a testing set (25%).

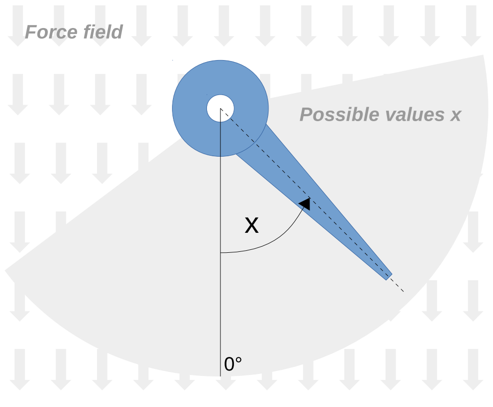


Figure 6.2: Simplified system

The discrete-states model is able to model any complicated pmf for transitions $p(\mathbf{s}_t|\mathbf{s}_{t-1})$, but as it requires a complete enumeration of possible state values, it becomes infeasible to train it for a larger set of possible values of states $\mathcal{S}$. Training the HMMs is studied in, e.g., [90]. It is straightforward to transform the discrete-state HMM into continuous-state version denoted as continuous-state hidden Gauss-Markov model (CS-HGMM). It requires to assume (multidimensional) Gaussian distribution of transition probability $p(\mathbf{s}_t|\mathbf{s}_{t-1})$, but the computational burden shrinks significantly. All the theoretical analysis provided above holds for the CS-HGMM as well and shows how it suits to the use case of robot modeling. For the experiments, we adopt these assumptions and we transform the discrete-state HMM into CS-HGMM. More details on this transformation is in the next Chapter 7 and appendix.

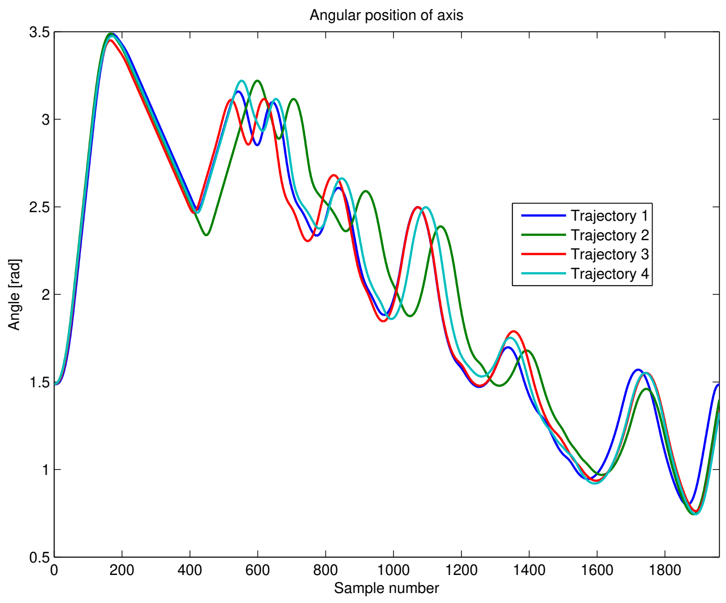


Figure 6.3: Angular position of axis for each measured trajectory.

6.3 Use Case Evaluation

Provided we have a CS-HGMM of the robotic robotic axis, we evaluate the classification performance and properties of the TSC framework proposed in this chapter.

Four different trajectories were measured to test the algorithm. As mentioned above 150 exemplary runs were measured for each trajectory to get a training set and then 50 runs to get a testing set. An example of the evolution of the axis angular position for each trajectory is shown on Figure 6.3.

A time-varying Markov model is trained from the training set using the frequency of occurrences of transitions as an approximate probability of transition. For computational advantages we assume Gaussian distributions in the model. For each of the four trajectories one Markov model is trained. The testing set is then used for evaluating the performance of the algorithm. This process is in fact a supervised learning from fully annotated data.

For the testing only the measurement of the power consumption is used. An example of consumption for the Trajectory 1 is given in Figure 6.5.

In each example the probability of one trajectory overgrows the rest. The probability of corresponding trajectories stabilises as more samples of measured sequence are taken in account.

The algorithm classifies correctly even such sequences that are very similar to each other. It can happen that a sequence from a certain class is distorted by noise so much that it is in fact more similar to a different class. In such a case it can be misclassified by the algorithm. To reduce these cases we introduced an additional criterion for classification. A given sequence can be classified only when its probability offset is bigger than a certain margin. Otherwise it is marked as undecided. These margins are subject to tuning for a particular set of trajectories. This way the risk of misclassification is drastically reduced.

In such a way we are not only able to classify the sequences but it is also possible to determine that a sequence cannot be classified reliably. This fact is worth pointing out because in such a case a further inspection (either manual or automatic by another algorithm) can be done (and recommended) based on the classification results. This may also be taken as a sign that something wrong has happened either in the system itself or in the measurement.

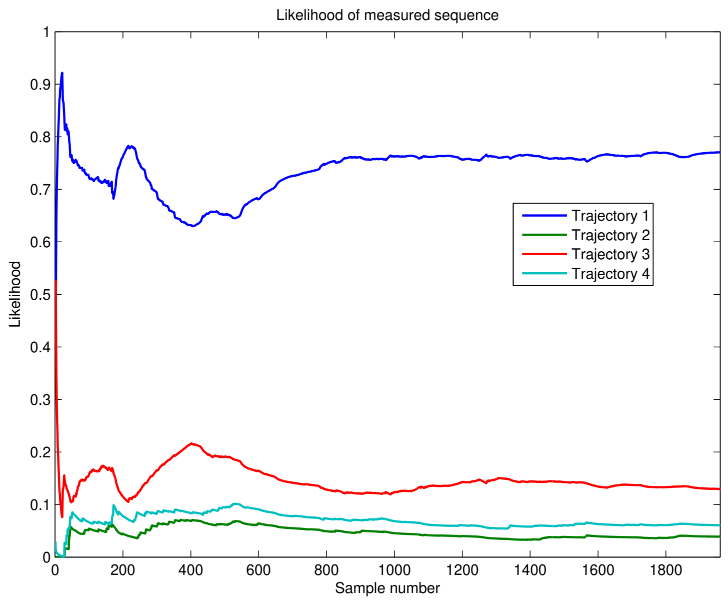


Figure 6.4: Evolution of probabilities of trajectories given measurements of Trajectory 1.

6.4 Conclusion

We have shown the capabilities of the stochastic models of dynamic systems. The reasoning about the likelihood of a model based on the observations and the structuring of the model into conditionally independent circuits justify the choice of this branch of models for this dissertation.

The structural approach we described in this chapter brings two important benefits. In contrast to the non-structural approach in Chapter 3, we obtain the implicit probability of the model. The probability is easy to interpret and normalization is done within the probabilistic framework. In the non-structural approach, additional calibration layers would have to be developed to bring the same level of interpretability.

The other benefit is that by having the Markov model of an input sequence separated, we can identify the robot dynamics independently in a separate experimental setup. A practical approach would be to identify the robot from experiments with known inputs and then use this part to identify tasks where inputs are unavailable. The input part of the model is fully stochastic and can be modeled separately.

We showed a general approach for probabilistic modeling of robot dynamics using discrete probabilities because to model is easier to set up. However, keeping the probabilities discrete leads to a high computational burden limiting the applications to relatively simple dynamics. To step up to the whole 6-DOF robotic manipulator, we have to use the continuous-state hidden Gauss-Markov models CS-HGMM. We elaborate more on this kind of models in the next chapter.

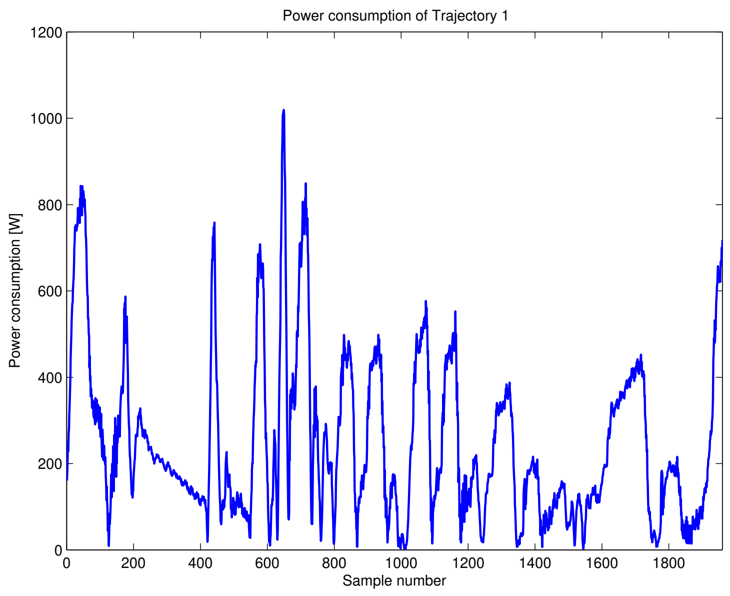


Figure 6.5: Example of a power consumption measurement used for classification.

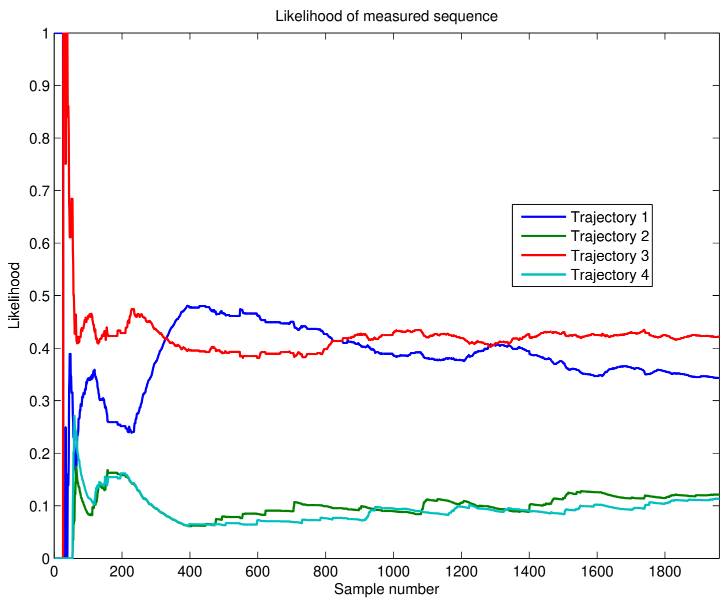


Figure 6.6: Worst case example of highly similar trajectories. Evolution of probabilities of trajectories given measurements of Trajectory 3.

Chapter 7

Continuity Preference in Parameters

In Chapter 6, we explored the path for modeling a 6-DOF industrial robotic manipulator as an example of a complex system. We laid down a basis in the form of analysis and experimental tests of HMM as a tool for modeling robotic operations. Both experiments and analysis are further deepened in this chapter, and a novel idea of continuity in parameters is presented. It turns out that the continuity preference significantly improves learning performance when only a limited amount of pattern examples are available. Additionally, the Expectation-Maximization EM algorithm (a classic learning algorithm) needs only a small augmentation to adopt the continuity preference. This is one of the main contributions of this dissertation work. The following text is based on our original work in [91].

7.1 Introduction

Smart manufacturing benefits from continuous monitoring of manufacturing operations. Such information provides essential knowledge to fine-tune the process control and to detect deviations in time. With flexible production, the scalability of a monitoring solution is one of the main concerns.

This chapter aims at detecting the current operational state of the robotic manipulators in the robotic line only from their power consumption. The monitoring system acts as a non-intrusive independent module, whose deployment requires only adding sensors at the power inlet of the robot. This way, deploying the monitoring system into running robotic cells is allowed without modifying the existing control systems.

Hidden Gauss-Markov models (HGMM) have been used extensively for the identification of systems, which possess the Markov property. That means no older states are considered given the state in the last time step [92]. Numerous examples can be found in the fields of natural language processing [93], [94], modeling of dynamic systems [95], image processing [96] and others. Unlike neural networks, the Gauss-Markov models are computationally favorable because the dynamic programming can be used. Standard industrial frameworks are available to run them in real-time. Moreover, the Gauss-Markov models inherently provide confidence in their estimates.

The idea of consistency proved useful in convolutional neural networks in the space domain, such as in [97], where the reconstruction of a depth map from a one-view video recording was solved without using a ground-truth depth map. In the case of HGMM, the consistency, which we define as the continuity preference, refers to the time domain.

The continuity preference can model a physical system continuous in its parameters. An example of such a system is an industrial manipulator (robot), a nonlinear continuous system that can be approximated with the Taylor series as a sequence of first-order linear systems. A resulting time-variant linear system can be modeled as a continuous-state time-varying hidden Gauss-Markov model (TV-HGMM).

An industrial robot executes multiple robotic programs, which we refer to as operations. Each operation consists of a sequence of trajectories that can follow instantly one after another or can be separated with an idle interval of an unknown length. Each trajectory is recorded as a sequence of power consumption measurements collected at the robot's power inlet. Each measurement can be either 1-dimensional, representing a single 3-phase power measurement, or multidimensional, where each dimension may represent a single measured quantity such as power at each phase of the inlet. The robotic programs are executed repeatedly during production. The time-varying parameters of the system capture the changing robot dynamics throughout the movement and the control inputs. Thus, the robot is viewed as an autonomous dynamic system whose total power consumption is measured at its inlet.

7.1.1 Objectives

The objective is to design a machine-learning algorithm for HGMM that allows for scalable, easy deployment in industrial systems. High discriminative power and capability to be generalized to various dataset types are the main requirements for the model to be used on various industrial systems. Another important aspect is learning from small datasets, which elevates the scalability of the solution. In the traditional learning of HGMM, the model robustness cannot be increased without increasing the dataset size. Requiring an extensive dataset negatively affects the ease of applicability. The continuity preference in HGMM utilizes the information from the time-adjacent samples, significantly increasing data utilization. A crucial part of the new algorithm is adapting the HGMM to consider the continuity in parameters. This approach can be applied to any other dynamic system that exhibits continuity in its parameters and shows repetitive behavior. Thanks to the HGMM discriminative power, even subtle and gradual changes in the system behavior are recognizable.

Another objective is to perform extensive testing both on benchmark and our use case data sets. Particular focus was given to the applicability in the industrial environment, especially for robotic manipulators. Their trajectories can be viewed as repetitive processes with unknown inputs and system parameters, observed only through fast-sampled power consumption measurements.

The final objective is to classify robotic operations in an online manner. This requirement disqualifies some highly accurate classifiers that are computationally too demanding. The HGMM satisfies this objective well.

7.1.2 Related Work

This chapter builds on the results published in [3], [10], [98]. Paper [10] approaches the identification of robotic operations using an external model of the direct measurement of individual robot power consumption. This approach has its limitations in automating the learning process, which highly reduces its usability in the industry to a bigger extent. Paper [3] models the robotic manipulator dynamics, making it possible to construct a practical method for semi-automatic learning of power consumption patterns. The topic has been further extended in [98], which deals with time series segmentation based on signal information contents, and presents unsupervised clustering of the acquired segments. The segmentation of robot trajectories was used as a prerequisite in the optimization tasks as described in [99], and [100], where the minimization of energy consumption of a robotic cell with two or more robots was achieved by changing the movement speeds of the individual robots.

The monitoring and prediction of manufacturing processes based only on non-intrusive methods have gained interest in smart buildings and smart electrical grids. In that field, it is called non-intrusive load monitoring (NILM). NILM aims to segregate the power consumption of particular electrical appliances from their aggregated power consumption. NILM offers to understand and predict the consumption based only on a single sensor measurement at the power inlet

[101], [102]. NILM assumes that each appliance model is represented by a simple on-off state (or possibly multiple states) with the mean power consumption of each state. Only inter-state transitions are allowed. These assumptions are not met in the case of continuously operating robotic manipulators, whose power consumption highly varies along the executed trajectory.

Another approach in robotics is described in [70], where the authors optimize power consumption of the robot movements without knowing the robot's precise internal structure. Their objective is to smooth out the robot motors' jerk based on knowing the existing trajectory obtained by gathering the position data from the robot joints during their movements in real-time.

Authors in [103] focus on fault detection and isolation in the industrial robotic manipulator based on its power consumption. They utilize the Bode Equations Vector Fitting to prepare healthy reference signatures of power consumption. Using a shallow neural network fed with data from joint positional encoders and total power consumption, the authors can detect and isolate faults in particular robot joints. This approach is promising, but it requires data from more sensors. We aim to be able to rely on a single sensor.

In the field of uncertain dynamics estimation for control, a classical form of neural networks showed to be useful in the adaptive estimation of uncertainties in models. In [104], authors employed the artificial neural network control technique for a flapping wing micro aerial vehicle. The neural network adapts the model uncertainties online during the flight based on the state estimate errors. Work of [105] derived a fuzzy neural network control for impedance control with constraints both in observed output and internal states of a robotic manipulator. The neural network fulfilled the role of an uncertain dynamics estimator. A simulation study on a 3-DOF manipulator was carried out, showing a promising tracking performance even when the simulated robot interacted with its environment. Paper [106] applies neural networks on fault detection in a 6-axis robot based on error prediction, but their approach requires accessibility of all joints to give predictions. All these approaches use the control input knowledge and require specific sensory data of the robot, which we aim to avoid.

A more general approach using the time-varying linear model of the process can solve a common problem of linear repetitive processes (LRP). Exploiting the repeating property of LRP, papers [107], [108] derive automatic learning of time-varying Kalman gain and noise covariances during the process execution. On the other hand, the classical approach of LRP assumes to have system matrices available for the learning phase. This assumption is rather demanding and is not necessary for our approach. Work of [109] investigates a repetitive process of a mobile robot operating along a typical trajectory using the Gaussian Process (GP) to estimate the locally constant dynamic parameters. In continuation [110], a different approach of Bayesian Linear Regression is chosen for estimating a part of unknown dynamics given a noisy input and a known model of the repeated process. The authors exploit the repetitiveness of the executed trajectory and estimate the hidden model across the repeated passes of the trajectory. In our approach, the repetitiveness is exploited too, but we do not assume to know any dynamics parameters.

Noteworthy is the approach of an unbiased minimum variance (MVU) estimation of state variables. This approach picks optimal state estimates so that the unbiased variance of simultaneously estimated input is minimal [111], [112]. However, this approach also requires the complete model of the system to be known.

From the works mentioned above, each solves only a part of the goal stated in Section 7.1.1. None of them solves the problem as a whole.

The work presented in this paper is based on the theoretical basis of unsupervised learning of HMMs as established, e.g., in [5], and on [113], which tackles the intricacies of constrained Expectation-Maximization learning. The results from these papers are extended to deal with the additional relaxed continuity constraint. The results from [98] are used for time-series segmentation to prepare the training datasets.

7.1.3 Chapter-Specific Notation

The topics from machine learning and dynamic system control fields overlap in this chapter, and consequently, some of the notation conventions differ. Special attention deserves notation for observations, dynamic state variables, and time series class labels. The following list describes the notation used in this chapter.

- Single measurement (observation) at a time instance $t \in \mathbb{R}$ within k-th sequence is a vector $\mathbf{y}_t^k \in \mathbb{R}^m$, corresponding hidden (latent) variable is a vector $\mathbf{x}_t^k \in \mathbb{R}^n$.
- k-th sequence of measurements (observations) from the first time instance to the final time instance T_k is
$$\mathbf{Y}_{T_k}^k = \left(\mathbf{y}_1^k, \mathbf{y}_2^k, ..., \mathbf{y}_{T_k}^k\right),$$
corresponding hidden (latent) variable sequence is
$$\mathbf{X}_{T_k}^k = \left(\mathbf{x}_1^k, \mathbf{x}_2^k, ..., \mathbf{x}_{T_k}^k\right).$$
- Set of all measurement sequences is denoted as
$$\mathcal{Y} = \{\mathbf{Y}_{T_1}^1, \mathbf{Y}_{T_2}^2, ...\mathbf{Y}_{T_k}^k, ..., \mathbf{Y}_{T_K}^K\},$$
set of all corresponding hidden variables sequences is
$$\mathcal{X} = \{\mathbf{X}_{T_1}^1, \mathbf{X}_{T_2}^2, ...\mathbf{X}_{T_k}^k, ..., \mathbf{X}_{T_K}^K\}.$$
- Hidden (latent) variables $\mathbf{x}_t^k$ used for statistical inference are equivalent to the state variables of the dynamic system.
- $\mathcal{C}$ denotes a set of all classes corresponding to distinctive robotic operations (programs).
- Whenever the superscript i is used, it denotes the i-th iteration of the Expectation Maximization (EM) algorithm run.
- A derivative of a (possibly matrix-valued) function $\mathbf{F}(\mathbf{A}_t)$ with respect to matrix $\mathbf{A}_t$ is denoted as
$$\mathcal{D}_{\mathbf{A}_t}\left\{\mathbf{F}(\mathbf{A}_t)\right\} = \frac{\partial \mathbf{F}(\mathbf{A}_t)}{\partial \mathbf{A}_t}.$$
- Vectorization of matrix denoted as $\mathrm{vec}(\mathbf{A})$ means stacking the columns of matrix $\mathbf{A}$ vertically forming a single-column vector.
- The Kronecker's product of matrices is denoted by the operator $\otimes$, e.g., $\mathbf{A} \otimes \mathbf{B}$.
- Operator $\mathbb{E}$ stands for the expected value. Subscript describes the respective probability distribution. E.g., expectation of some function $q(\mathcal{X})$ w.r.t. a distribution $p(\mathcal{X}|\mathcal{Y})$ is denoted as
$$\mathbb{E}_{\mathcal{X}|\mathcal{Y}}\left[q(\mathcal{X})\right] = \int p(\mathcal{X}|\mathcal{Y})q(\mathcal{X})d\mathcal{X}.$$

7.2 Learning of Structured Model

7.2.1 Dynamic Model of Robotic Manipulator

For gray-box modeling, analysis of the dynamic model of the system is needed. In the case of robotic manipulators, there is a well-founded basis for modeling their dynamics in the form of an actuated serial chain such as shown, e.g., in [6]–[8]. Robot KUKA KR5 being used in this project was modeled in [9]. For details on the structured model, see the Chapter 2.

The dynamics of a serial chain manipulator with n_r links can be described as:

$$\boldsymbol{\tau} = \mathbf{H}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}), \tag{7.1}$$

where $\mathbf{q} \in \mathbb{R}^{n_r}$ is the vector of joints displacements, $\boldsymbol{\tau} \in \mathbb{R}^{n_r}$ is a vector of joints' torques, $\mathbf{H}(\mathbf{q}) \in \mathbb{R}^{n_r \times n_r}$ is the inertia matrix, $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) \in \mathbb{R}^{n_r \times n_r}$ is a matrix of Coriolis and centrifugal forces and $\mathbf{g}(\mathbf{q}) \in \mathbb{R}^{n_r}$ is representing gravity force and other external forces exerted on the manipulator. The elements of matrices $\mathbf{H}(\mathbf{q})$, $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})$, and vector $\mathbf{g}(\mathbf{q})$ are composed of continuous functions [7].

The manipulator dynamics is a non-linear function. The EM algorithm searches for a piecewise linear approximation of equation (7.1) along a repetitive trajectory. This can be viewed as a first-order Taylor series approximation. The linearization may cause a residual error here, but experiments in Section 7.4 show that this error is acceptable.

The system is assumed to be autonomous so that the control inputs are modeled as a part of the system's dynamics. This way, state-space feedback control is assumed, but the robot may also be controlled via separate motor feedback loops (e.g., proportional-integral controller). The control parameters are not known to a robot's user, which generates uncertainty. This uncertainty is learned during training. Using model (7.1), the formal expression of a discrete state-space model that is being searched for is

$$\mathbf{x}_t = \mathbf{A}_t\mathbf{x}_{t-1} + \mathbf{v}_t, \mathbf{v}_t \sim \mathcal{N}(0, \mathbf{Q}_t), \tag{7.2}$$

$$\mathbf{y}_t = \mathbf{B}\mathbf{x}_t + \mathbf{e}_t, \mathbf{e}_t \sim \mathcal{N}(0, \mathbf{R}), \tag{7.3}$$

where $\mathbf{x}_{t+1}, \mathbf{x}_t \in \mathbb{R}^n$ are vectors of states of the system in respective time instances. $\mathbf{A}_t \in \mathbb{R}^{n\times n}$ is a state evolution matrix of the piecewise linearly approximated system at time instance t. Vectors $\mathbf{v}_t \in \mathbb{R}^n, \mathbf{e}_t \in \mathbb{R}^m$ are process and measurement noise respectively, which are both assumed here to be normal i.i.d random variables. This assumption is usually used in theory as deterministic correlations within the noise can be modeled as an additional dynamics in the $\mathbf{A}_t$ matrix practically lowering the effect of potentially neglecting correlations in the noise. $\mathbf{y}_t \in \mathbb{R}^m$ is an output measurement vector and $\mathbf{B} \in \mathbb{R}^{m\times n}$ is an observation matrix.

The need for automatic estimation of the Markov model parameters emerges from the high complexity of the modeled system. In practice, the first three axes of the robot are the primary power consumers and given only the power consumption measurement, the analysis of the dynamics suggests using six hidden states. It turns out from our experiments that three hidden states perform sufficiently well.

Markov models parameters are known to be successfully estimated by the Baum-Welch algorithm, which belongs to the EM algorithms family. Alternatively, the problem of selecting the number of hidden states or generally the model selection can be approached using the Akaike or Bayesian information criterion (AIC, BIC) as shown, e.g., in [114]–[116]. This article uses a continuous-state time-varying version of the hidden Gauss-Markov model (abbreviated here as TV-HGMM). A modification of the EM algorithm is applied for unsupervised learning of the model and enhanced to reflect the continuity in parameters which is suggested by the analysis of parameters in equation (7.1) describing the physical system. The continuity property naturally occurs in many kinds of physical systems.

7.2.2 Formal Definition

Let the physical system from (7.2), (7.3) be described in the form of a stochastic model:

$$\begin{aligned} p(\mathbf{x}_1; \mathbf{\Theta}_1) &= \mathcal{N}(\mathbf{x}_1; \boldsymbol{\mu}_1, \mathbf{P}_1), \\ p(\mathbf{x}_t|\mathbf{x}_{t-1}; \mathbf{\Theta}_X) &= \mathcal{N}(\mathbf{x}_t; \mathbf{A}_t\mathbf{x}_{t-1}, \mathbf{Q}_t), \\ p(\mathbf{y}_t|\mathbf{x}_t; \mathbf{\Theta}_Y) &= \mathcal{N}(\mathbf{y}_t; \mathbf{B}\mathbf{x}_t, \mathbf{R}), \end{aligned}$$

where $\mathbf{\Theta}$ represents the respective sets of parameters of distributions, $\mathcal{N}(.)$ denotes a multivariate normal distribution. Covariance matrices $\mathbf{P}_1$, $\mathbf{Q}_t \in \mathbb{R}^{n\times n}$ and $\mathbf{R} \in \mathbb{R}^{m\times m}$ are assumed to be symmetric positive definite. $\mathbf{A}_t$ is a system time-evolution (transition) matrix, $\mathbf{B}$ is an output measurement matrix, $\boldsymbol{\mu}_1$ and $\mathbf{P}_1$ are the mean and covariance of the initial state $\mathbf{x}_1$, respectively. $\mathbf{Q}_t$ and $\mathbf{R}$ are covariance matrices of the system noise $\mathbf{v}_t$ and measurement noise $\mathbf{e}_t$, respectively. Standard Kalman-filter subscripts are used further in the paper.

Note that the output measurement model represented by the matrix $\mathbf{B}$ is assumed to be time-invariant because the measurement model's temporal changes are negligible. All the temporal system variations are captured by the system matrix $\mathbf{A}_t$, which is time-varying. Nevertheless, it is straightforward to generalize our approach for a time-varying output model too.

The objective function J for learning is the marginal likelihood of the parameters $\mathbf{\Theta}$ given the observed data $\mathcal{Y}$. Formally,

$$J(\mathcal{X}, \mathcal{Y}, \mathbf{\Theta}) = p(\mathcal{Y}; \mathbf{\Theta}) = \int p(\mathcal{X}, \mathcal{Y}; \mathbf{\Theta}) d\mathcal{X}.$$

Optimizing the objective in this form analytically is intractable. The formula can be lower-bounded by a proxy objective as

$$\begin{aligned} \ln p(\mathcal{Y}; \mathbf{\Theta}) &= \ln \int p(\mathcal{X}, \mathcal{Y}; \mathbf{\Theta}) d\mathcal{X} \\ &= \ln \int p(\mathcal{X}, \mathcal{Y}; \mathbf{\Theta}) \frac{p(\mathcal{X}|\mathcal{Y}; \mathbf{\Theta})}{p(\mathcal{X}|\mathcal{Y}; \mathbf{\Theta})} d\mathcal{X} \\ &= \ln \left(\mathbb{E}_{\mathcal{X}|\mathcal{Y};\mathbf{\Theta}} \left[\frac{p(\mathcal{X}, \mathcal{Y}; \mathbf{\Theta})}{p(\mathcal{X}|\mathcal{Y}; \mathbf{\Theta})} \right] \right) \qquad (7.4) \\ &\geq \mathbb{E}_{\mathcal{X}|\mathcal{Y};\mathbf{\Theta}} \left[\ln p(\mathcal{Y}, \mathcal{X}; \mathbf{\Theta}) \right] \\ &\quad - \mathbb{E}_{\mathcal{X}|\mathcal{Y};\mathbf{\Theta}} \left[\ln p(\mathcal{X}|\mathcal{Y}; \mathbf{\Theta}) \right], \qquad (7.5) \end{aligned}$$

where Jensen's inequality was applied in transition from step (7.4) to (7.5). It can be shown, that maximizing only the expectation term

$$\mathbb{E}_{\mathcal{X}|\mathcal{Y};\mathbf{\Theta}} \left[\ln p(\mathcal{Y}, \mathcal{X}; \mathbf{\Theta}) \right] \qquad (7.6)$$

from (7.5) leads to maximizing the original lower bound (7.5) [117], leading to the optimization task:

$$\mathbf{\Theta}^* = \arg\max_{\mathbf{\Theta}} \mathbb{E}_{\mathcal{X}|\mathcal{Y};\mathbf{\Theta}} \left[\ln p(\mathcal{Y}, \mathcal{X}; \mathbf{\Theta}) \right]. \qquad (7.7)$$

By maximizing the lower bound (7.6), a close approximation of the optimum is attained. However, the closed-form solution for (7.7) becomes intractable quickly with the growing number of training samples. Numerical optimization techniques can be applied, and in the field of dynamic system control, the EM algorithm is often the choice for this task. It is well studied in its application on exponential families of distribution (e.g., [5], [117]). Its extension in this paper turns out to be relatively easy to implement.

7.2.3 EM Learning

We will call the term (7.6) the auxiliary function $Q(\boldsymbol{\Theta}, \boldsymbol{\Theta}^i)$ as in [5]. Note that its parameter set is explicitly partitioned for alternating steps. Formally,

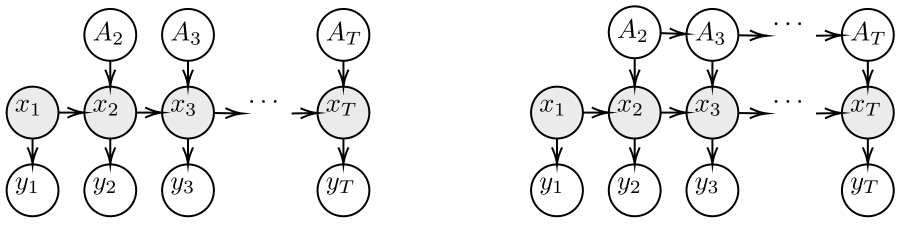
$$Q(\boldsymbol{\Theta}, \boldsymbol{\Theta}^i) = \mathbb{E}_{\mathcal{X}|\mathcal{Y};\boldsymbol{\Theta}^i}\left[\ln p(\mathcal{Y}, \mathcal{X}; \boldsymbol{\Theta})\right]. \tag{7.8}$$

The EM algorithm estimates parameters of a probability distribution function (pdf) iteratively in two steps. The expectation step (E-step) calculates the hidden states' expected values given the last estimate of the pdf parameters and the measurement sequence. The Maximization step (M-step) follows, which estimates the pdf parameters given the freshly inferred hidden states from the E-step. That is done by maximizing the likelihood of the parameters given the data. These two steps alternate until convergence into the local optimum. A detailed description and complete formulas can be found in A.2.

7.2.4 Parameter Continuity Preference

It is often the case that a huge amount of training data is required, even for the time-invariant HGMM. This data necessity amplifies significantly in the case of the time-varying HGMM. Another issue with time-varying models is that the input data must be timed appropriately. The starts of the training sequences must be consistently and precisely repeated. Both of these issues discourage practitioners from using these models. However, the time-varying models show a better performance in regression tasks and classification tasks, as shown in the experimental Subsection 7.4.3.

This article proposes a solution to these issues. Assume that the system matrices $\mathbf{A}_t$ are random matrices generated from a multivariate Wiener process. In other words, the system matrices $\mathbf{A}_t$ are sampled from a continuous stochastic process with normally distributed increments between samples. The continuity condition reflects the expected properties of dynamic parameters of a serial robotic manipulator described, e.g., in [118], and in Subsection 7.2.1. The continuity assumption results in a significant improvement in training the time-varying HGMM and reduces the overfitting. The intuition behind that is visualized in Figure 7.1. It is an extension of the idea of dependency of time-adjacent hidden states. Similarly, the time-adjacent parameters should also be dependent on each other. Moreover, since we encourage the model to tolerate more significant deviations from the expected observation, most deviations accumulate only in system noise and measurement noise covariances $\mathbf{Q}_t$ and $\mathbf{R}$.



(a) Plain HMM (b) Continuity preference HMM

Figure 7.1: HMM structures. Comparison between the plain structure of HMM and the continuity constrained HMM in the form of dependency Bayes net. The difference is in the additional dependence between the nodes $\mathbf{A}_t$.

To formalize, let the difference of the system matrices be defined as

$$\tilde{\mathbf{A}}_t = \mathbf{A}_t - \mathbf{A}_{t-1}.$$

Suppose that the vectorized difference of the system matrices is a normally distributed random variable with zero mean and a diagonal covariance forming pdf

$$\operatorname{vec}(\tilde{\mathbf{A}}_t) \sim \mathcal{N}\left(\operatorname{vec}(\tilde{\mathbf{A}}_t); \mathbf{0}, \frac{1}{\kappa}\mathbf{I}\right), \operatorname{vec}(\tilde{\mathbf{A}}_t) \in \mathbb{R}^{n^2}, \kappa \in \mathbb{R}^+.$$

The auxiliary function $Q(\mathbf{\Theta}, \mathbf{\Theta}^i)$ from (7.8) can be extended by the term Q_A forming the augmented auxiliary function $Q_a(\mathbf{\Theta}, \mathbf{\Theta}^i)$ formally defined as

$$Q_a(\mathbf{\Theta}, \mathbf{\Theta}^i) = \mathbb{E}_{\mathcal{X}|\mathcal{Y};\mathbf{\Theta}^i} \left[\ln p(\mathcal{Y}, \mathcal{X}, \tilde{\mathbf{A}}_t; \mathbf{\Theta}) \right]. \tag{7.9}$$

The optimization task becomes

$$\mathbf{\Theta}^{i+1} = \arg\max_{\mathbf{\Theta}} Q_a(\mathbf{\Theta}, \mathbf{\Theta}^i), \tag{7.10}$$

and the auxiliary function can be decomposed as

$$\begin{aligned} Q_a(\mathbf{\Theta}, \mathbf{\Theta}^i) &= Q(\mathbf{\Theta}, \mathbf{\Theta}^i) + Q_A \\ &= Q_1 + Q_X + Q_Y + Q_A, \end{aligned} \tag{7.11}$$

and each term is maximized separately. Q_1, Q_X and Q_Y are defined in (A.23)-(A.25). Given that $\text{vec}(\tilde{\mathbf{A}}_t)$ is a random variable independent of $\mathcal{X}$ and by using lemma (3), the augmentation part Q_A is defined as

$$\begin{aligned} Q_A &= -\frac{1}{2}\left(Z(\kappa) + \kappa \sum_{t=3}^{T} \Psi_t \right), \\ Z(\kappa) &= (T-1)(n^2 \ln(2\pi) - \ln \kappa), \end{aligned}$$

where Ψ_t is the squared Frobenius norm of the matrix difference $\tilde{\mathbf{A}}_t$ defined as

$$\Psi_t = ||\tilde{\mathbf{A}}_t||_F^2 = \text{tr}(\tilde{\mathbf{A}}_t^\top \tilde{\mathbf{A}}_t) = \text{vec}(\tilde{\mathbf{A}}_t)^\top \text{vec}(\tilde{\mathbf{A}}_t).$$

From this substitution, one can view the augmentation also as a demand that the distance of consecutive matrices $\mathbf{A}_t$, $\mathbf{A}_{t-1}$ is penalized. This penalty keeps the matrices close to each other. The κ coefficient controls the amount of penalization.

In a sense, the parameter κ represents a mixing of the TV-HGMM and TI-HGMM. As κ goes to zero, the resulting model becomes the pure TV-HGMM. On the other hand, as the κ goes to infinity, the resulting model becomes the pure TI-HGMM or Kalman filter. The effect of this idea is visualized in Figure 7.2.

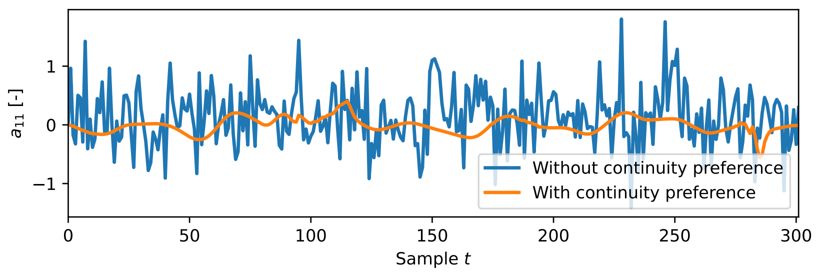


Figure 7.2: Continuity preference effect. Example of evolution of a single element a_{11} of the time-varying system matrix $\mathbf{A}_t$. The continuity preference induces a significantly smoother evolution as the time-adjacent matrices are similar. This evolution comes from the model of KR210 dataset.

7.2.5 M-step Augmented by Continuity Preference

The continuity preference can be incorporated into the model by any optimization technique that solves task (7.10), and the interpretation will hold. In this subsection, we derive the M-step's augmentation from the EM algorithm with respect to the specifics of the continuity preference. It turns out that only a simple additional modification of the EM algorithm is needed.

To find the optimal $\mathbf{A}_t$ as a part of task (7.10), one must solve the differential equation

$$\mathcal{D}_{\mathbf{A}_t}\left\{Q_a(\boldsymbol{\Theta},\boldsymbol{\Theta}^i)\right\} = \mathcal{D}_{\mathbf{A}_t}\left\{Q(\boldsymbol{\Theta},\boldsymbol{\Theta}^i)\right\} + \mathcal{D}_{\mathbf{A}_t}\left\{Q_A\right\} = 0$$

for $\mathbf{A}_t$, where the individual terms are

$$\begin{aligned}\mathcal{D}_{\mathbf{A}_t}\left\{Q(\boldsymbol{\Theta},\boldsymbol{\Theta}^i)\right\} &= \mathbf{Q}_t^{-1}\mathbf{A}_t\mathbf{C}_{\mathbf{x}_{t-1}\mathbf{x}_{t-1}} - \mathbf{Q}_t^{-1}\mathbf{C}_{\mathbf{x}_t\mathbf{x}_{t-1}}, &(7.12)\\ \mathcal{D}_{\mathbf{A}_t}\left\{Q_A\right\} &= \frac{\kappa}{2}\mathcal{D}_{\mathbf{A}_t}\left\{\sum_{t=3}^{T}\Psi_t\right\}, &(7.13)\end{aligned}$$

where the correlation statistics are defined as

$$\begin{aligned}\mathbf{C}_{\mathbf{x}_t\mathbf{x}_t} &= \frac{1}{K_t}\sum_{k=1}^{K_t}\left(\mathbf{P}_{t|T_k} + \boldsymbol{\mu}_{t|T_k}^k\boldsymbol{\mu}_{t|T_k}^{k\top}\right), &(7.14)\\ \mathbf{C}_{\mathbf{x}_t\mathbf{x}_{t-1}} &= \frac{1}{K_t}\sum_{k=1}^{K_t}\left(\mathbf{P}_{t|T_k}\mathbf{H}_t + \boldsymbol{\mu}_{t|T_k}^k\boldsymbol{\mu}_{t-1|T_k}^{k\top}\right), &(7.15)\end{aligned}$$

where $\mathbf{H}_t$ comes from Equation (A.18) of the E-step.

$$\begin{aligned}\mathcal{D}_{\mathbf{A}_t}\left\{\sum_{t=3}^{T-1}\Psi_t\right\} &= \mathcal{D}_{\mathbf{A}_t}\{\operatorname{tr}(\mathbf{A}_t^\top\mathbf{A}_t) - 2\operatorname{tr}(\mathbf{A}_{t-1}^\top\mathbf{A}_t)\\ &\quad + \operatorname{tr}(\mathbf{A}_{t-1}^\top\mathbf{A}_{t-1}) + \operatorname{tr}(\mathbf{A}_{t+1}^\top\mathbf{A}_{t+1})\\ &\quad -2\operatorname{tr}(\mathbf{A}_{t+1}^\top\mathbf{A}_t) + \operatorname{tr}(\mathbf{A}_t^\top\mathbf{A}_t)\}\\ &= 4\mathbf{A}_t - 2\mathbf{A}_{t+1} - 2\mathbf{A}_{t-1}. &(7.16)\end{aligned}$$

The auxiliary function Ψ_t depends on $\mathbf{A}_t$ in two consequent time-steps in the entire sum, while the edge cases $\mathbf{A}_2, \mathbf{A}_T$ are expressed separately as

$$\begin{aligned}\mathcal{D}_{\mathbf{A}_2}\{\Psi_3\} &= \mathcal{D}_{\mathbf{A}_2}\{\operatorname{tr}(\mathbf{A}_3^\top\mathbf{A}_3) - 2\operatorname{tr}(\mathbf{A}_2^\top\mathbf{A}_3)\\ &\quad + \operatorname{tr}(\mathbf{A}_2^\top\mathbf{A}_2)\}\\ &= 2\mathbf{A}_2 - 2\mathbf{A}_3, &(7.17)\\ \mathcal{D}_{\mathbf{A}_T}\{\Psi_T\} &= \mathcal{D}_{\mathbf{A}_T}\{\operatorname{tr}(\mathbf{A}_T^\top\mathbf{A}_T) - 2\operatorname{tr}(\mathbf{A}_{T-1}^\top\mathbf{A}_T)\\ &\quad + \operatorname{tr}(\mathbf{A}_{T-1}^\top\mathbf{A}_{T-1})\}\\ &= 2\mathbf{A}_T - 2\mathbf{A}_{T-1}. &(7.18)\end{aligned}$$

Plugging expressions (7.16), (7.17), (7.18) into (7.13) and equaling the derivative to zero, we get

$$\begin{aligned}0 &= \mathbf{Q}_t^{-1}\mathbf{A}_t\mathbf{C}_{\mathbf{x}_{t-1}\mathbf{x}_{t-1}} - \mathbf{Q}_t^{-1}\mathbf{C}_{\mathbf{x}_t\mathbf{x}_{t-1}}\\ &\quad +\kappa\left(2\mathbf{A}_t - \mathbf{A}_{t+1} - \mathbf{A}_{t-1}\right), &(7.19)\\ 0 &= \mathbf{Q}_2^{-1}\mathbf{A}_2\mathbf{C}_{\mathbf{x}_1\mathbf{x}_1} - \mathbf{Q}_2^{-1}\mathbf{C}_{\mathbf{x}_2\mathbf{x}_1}\\ &\quad +\kappa\left(\mathbf{A}_1 - \mathbf{A}_2\right), &(7.20)\\ 0 &= \mathbf{Q}_T^{-1}\mathbf{A}_T\mathbf{C}_{\mathbf{x}_{T-1}\mathbf{x}_{T-1}} - \mathbf{Q}_T^{-1}\mathbf{C}_{\mathbf{x}_T\mathbf{x}_{T-1}}\\ &\quad +\kappa\left(\mathbf{A}_T - \mathbf{A}_{T-1}\right). &(7.21)\end{aligned}$$

We want to solve these equations for $\mathbf{A}_t$. Rearranging and using formula

$$\operatorname{vec}(\mathbf{AXB})^\top = \operatorname{vec}(\mathbf{X})^\top(\mathbf{B}\otimes\mathbf{A}^\top), \tag{7.22}$$

where $\otimes$ stands for the Kronecker's product, we get the final analytical expression

$$\begin{aligned}\mathrm{vec}(\mathbf{A}_t)^\top &= \mathrm{vec}(\mathbf{C}_{\mathbf{x}_t\mathbf{x}_{t-1}} + \kappa\mathbf{Q}_t(\mathbf{A}_{t-1} + \mathbf{A}_{t+1}))^\top \\ &\quad \times \left[(\mathbf{C}_{\mathbf{x}_{t-1}\mathbf{x}_{t-1}} \otimes \mathbf{I}_n) + 2\kappa(\mathbf{I}_n \otimes \mathbf{Q}_t)\right]^{-1}. \end{aligned} \tag{7.23}$$

From $\mathrm{vec}(\mathbf{A}_t)$, the desired matrix $\mathbf{A}_t$ is obtained by reshaping in the column-major manner into an $n \times n$ matrix. Solving (7.23) directly is prone to numerical issues. An alternative approach is to rearrange (7.19), (7.20), and (7.21) into Sylvester equations

$$2\kappa\mathbf{Q}_t\mathbf{A}_t + \mathbf{A}_t\mathbf{C}_{\mathbf{x}_{t-1}\mathbf{x}_{t-1}} = \mathbf{C}_{\mathbf{x}_t\mathbf{x}_{t-1}} + \kappa\mathbf{Q}_t(\mathbf{A}_{t-1} + \mathbf{A}_{t+1}), \tag{7.24}$$

$$\kappa\mathbf{Q}_2\mathbf{A}_2 + \mathbf{A}_2\mathbf{C}_{\mathbf{x}_1\mathbf{x}_0} = \mathbf{C}_{\mathbf{x}_2\mathbf{x}_1} + \kappa\mathbf{Q}_2\mathbf{A}_3, \tag{7.25}$$

$$\kappa\mathbf{Q}_T\mathbf{A}_T + \mathbf{A}_T\mathbf{C}_{\mathbf{x}_{T-1}\mathbf{x}_{T-1}} = \mathbf{C}_{\mathbf{x}_T\mathbf{x}_{T-1}} + \kappa\mathbf{Q}_T\mathbf{A}_{T-1}, \tag{7.26}$$

and solve (7.24) for $\mathbf{A}_t$, (7.25) for $\mathbf{A}_2$, and (7.26) for $\mathbf{A}_T$. Numerically robust solvers for Sylvester equations are available.

Since matrix $\mathbf{A}_t$ depends not only on preceding matrix $\mathbf{A}_{t-1}$, but also on next matrix $\mathbf{A}_{t+1}$, which is not available during the forward pass of the estimation, we need an expression to get the estimate of $\mathbf{A}_{t+1}$ at time t. We use solution (A.40) obtained by optimizing the original $Q(\boldsymbol{\Theta}, \boldsymbol{\Theta}^i)$ function.

$$\mathbf{A}_{t+1} = \mathbf{C}_{\mathbf{x}_{t+1}\mathbf{x}_t}\mathbf{C}^{-1}_{\mathbf{x}_t\mathbf{x}_t}. \tag{7.27}$$

The parameters for the density of the first sample remain the same as in (A.44)-(A.45) and are expressed as

$$\boldsymbol{\mu}_1^{i+1} = \frac{1}{K}\sum_{k=1}^{K}\boldsymbol{\mu}^k_{1|T_k}, \tag{7.28}$$

$$\boldsymbol{\varepsilon}_1^{k,i+1} = \boldsymbol{\mu}^k_{1|T_k} - \boldsymbol{\mu}_1^{i+1}, \tag{7.29}$$

$$\mathbf{P}_1^{i+1} = \frac{1}{K}\sum_{k=1}^{K}\left(\mathbf{P}_{1|T_k} + \boldsymbol{\varepsilon}_1^{k,i+1}\boldsymbol{\varepsilon}_1^{k,i+1^\top}\right). \tag{7.30}$$

Note that superscript k denotes an index of the particular k-th training sequence while superscript i denotes i-th iteration of the EM algorithm. K is the total number of training sequences. For details about standard EM algorithm derivation, see A.2.

As the reader can see, the Equations (7.24), (7.25), (7.26) are the only modification to the standard EM algorithm that is needed to impose continuity preference on the parameters of the model.

The continuity preference demands an additional computational load of each EM iteration. For a standard EM, each iteration has computational complexity $\mathcal{O}(Tn^3)$ and solving the Sylvester equations has complexity $\mathcal{O}(Tn^3)$, see [119], [120]. Thus, the overall computational complexity remains $\mathcal{O}(Tn^3)$ per EM iteration, which is linear in the number of data samples. Note that the value of κ does not further affect the computational load.

7.2.6 Initialization of EM algorithm

The EM algorithm ends up in a local optimum as it optimizes the lower bound (7.6). That means it is sensitive to the initialization of model parameters. Another issue is the numerical stability of steps during the matrix inversions when an ill-conditioned matrix arises.

The numerical issues are mitigated here using approaches for parameter initialization described in [121]. Initialization issues are subjected to a detailed discussion in the literature, e.g., [122]. The local optimum issue can be tackled either by some system identification task, white-box, gray-box identification, or repeated trials with random initialization.

The random initialization performed best during our experiments. It is sufficient to initialize the hidden states with bounded values and begin learning with the M-step. Immediately in the first M-step, the parameters are numerically stable. This approach is highly recommended for practitioners as it minimizes the workload required for initialization.

7.3 Classification of Operations

The classification of operations is the practical outcome of this paper, built on the models trained with the continuity preference as described in Section 7.2. Thus, once the hidden Gauss-Markov model (HGMM) is trained for each distinctive robotic operation, it is used to evaluate the likelihood of an operation given a series of measurements. The likelihood can be calculated either offline using the realization of the whole run of the E-step described in A.2.1 or online using only the forward recursion. The overview of the classification pipeline is depicted in Figure 7.3

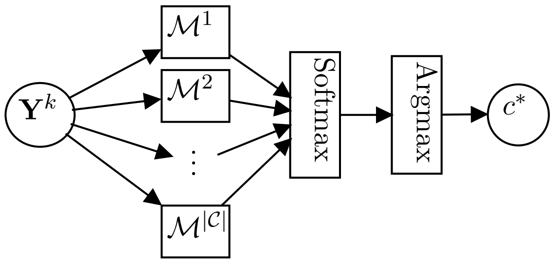


Figure 7.3: Classification pipeline design. The measurement is copied into each of the models $\mathcal{M}^c$.

Suppose superscript $c \in \mathcal{C}$ denotes a distinctive robotic operation, which can also be called a class from the classification point of view. Such an operation can be, e.g., pick-and-place of a manufactured part or a movement when the robot holds a part and applies glue on it at a gluing station. Also, let $\mathbf{Y}^k = \left(\mathbf{y}_1^k, \mathbf{y}_2^k, ..., \mathbf{y}_T^k\right)$ denote the k-th measured sequence to be classified. The predictive density of the model $\mathcal{M}^c$ of a class c is evaluated at each measurement sample $\mathbf{y}_t^k$, where:

$$\mathbf{y}_t^k \sim p(\mathbf{y}_t^k|\mathcal{M}^c) = \mathcal{N}(\mathbf{y}_t^k; \mathbf{B}^c \boldsymbol{\mu}_{t|t-1}^c, \boldsymbol{\Sigma}_t^c), \tag{7.31}$$

where $\boldsymbol{\Sigma}_t^c$ is expressed in (A.14) and $\boldsymbol{\mu}_{t|t-1}^c$ in (A.13).

For classification, we use the log point-wise predictive density of the model $\mathcal{M}^c$ evaluated for the whole sequence $\mathbf{Y}^k$ as [123]:

$$\begin{aligned} lpd(\mathbf{Y}^k|\mathcal{M}^c) &= \ln p(\mathbf{Y}^k|\mathcal{M}^c) \\ &= \frac{1}{T}\sum_{t=1}^{T} \ln \mathcal{N}(\mathbf{y}_t^k; \mathbf{B}^c \boldsymbol{\mu}_{t|t-1}^c, \boldsymbol{\Sigma}_t^c). \end{aligned} \tag{7.32}$$

The resulting *lpd*s are fed into the weighted log-softmax calibration layer to yield appropriate estimates of class probabilities. The dataset $\mathcal{Y}$ was stratified by respective classes forming a total of $|\mathcal{C}|$ datasets $\mathcal{Y}^c$. The stratified datasets $\mathcal{Y}^c$ were partitioned into the training sets $\mathcal{Y}_{tr}^c$, and the validation sets $\mathcal{Y}_v^c$. 30 % of samples were held in the validation sets $\mathcal{Y}_v^c$.

The weighted log-softmax function follows the expression (because of the numerical stability, the logarithmic transformation is used):

$$\begin{aligned} \sigma_{log}(z_c, \mathbf{w}) &= \log\left(\frac{\exp(w_c z_c)}{\sum_{i \in C} \exp(w_i z_i)}\right), \\ z_c &= z_c(\mathbf{Y}^k) = lpd(\mathbf{Y}^k|\mathcal{M}^c), \end{aligned}$$

where $\mathbf{w} = [w_1, w_2, ..., w_{|C|}]^\top$ is the calibration weighting vector obtained by minimizing the cross-entropy:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_{c \in C} \sum_{\mathbf{Y}^k \in \mathcal{Y}^c_{tr}} \sigma_{log}\left(lpd(\mathbf{Y}^k|\mathcal{M}^c), \mathbf{w}\right).$$

The class' score $lpd(\mathbf{Y}^k|\mathcal{M}^c)$ is evaluated for each of the models. The classification task is then defined as selecting the class with the highest probability, formally as:

$$c^* = \arg\max_c \sigma_{log}(z_c, \mathbf{w}), \tag{7.33}$$

where c^* represents the maximum probability class.

The classification task can be carried out online in a recurrent manner. In such a case, the recurrent evaluation of (7.33) can be viewed as an any-time algorithm improving its accuracy with each consecutive measurement.

7.4 Experiments and Results

The learning algorithm was tested on three types of datasets. These datasets are published online [124]. The first type is a public benchmark dataset preprocessed for the time series classification task. The second type is the power consumption of a real robot in laboratory conditions, whose trajectories are very similar to each other. The third type is the power consumption measured on a real robot in a production line in the car body shop in Skoda Auto during live production, where the trajectories differ a lot, but the number of training examples in classes is imbalanced. The power consumption was measured at the inlet of the robot.

As a baseline for comparing performance, the standard versions of TI-HGMM and TV-HGMM are trained and evaluated. In the classification task, we also enlist the performance of the three common benchmark time series classifiers KNN (k nearest neighbors with Euclidean distance measure), KNN-DTW (KNN with the dynamic time-warping distance measure), and the BOSSE (Bag of SFA Symbols Ensemble).

We propose two metrics based on predictive density to compare performance and add them to the standard accuracy and cross-entropy loss classification metrics. A detailed description of those metrics is in the following section.

The experiments emphasize the main advantage of the continuity preference approach, which is the capability of the algorithm to learn from only a few training examples. The main interest resides in showing the effect of different settings of the continuity parameter κ. We applied the method of Bayesian optimization [125] to search the space of κ for the optimal values instead of a less efficient random search. In the Pareto-optimization types of experiments, we run ten trials of each experiment, applying the leave-one-out cross-validation scheme per trial. Averaged results are depicted in Figures 7.5a, 7.5c, and 7.5b.

7.4.1 Applied Methods and Metrics

The parameter κ scales quadratically with its effect of smoothing the evolution matrix $\mathbf{A}_t$ as it is a variation of a probability distribution. Therefore, we write it as a squared number to improve readability.

For the model performance metric, the expected log point-wise predictive density $\widehat{elpd}$ from [123] was used. This metric indicates the predictive accuracy of the regression model and considers the confidence in its predictions: the higher $elpd$, the better performance of the model. The true $elpd$ can only be approximated. Here it was estimated using the leave-one-out cross-validation. For each fold, each class dataset is partitioned into a training set and a validation set, i.e., $\mathcal{Y}^c = \mathcal{Y}^c_{tr} \cup \mathcal{Y}^c_v$, disjoint so that $\mathcal{Y}^c_{tr} \cap \mathcal{Y}^c_v = \varnothing$. The estimate $\widehat{elpd}$ for model $\mathcal{M}^c$ of class c trained on the subset $\mathcal{Y}^c_{tr}$ is then defined as:

$$\widehat{elpd}(\mathcal{Y}^c_v, \mathcal{M}^c) = \frac{1}{|\mathcal{Y}^c_v|} \sum_{\mathbf{Y}^k \in \mathcal{Y}^c_v} \ln p(\mathbf{Y}^k | \mathcal{M}^c), \tag{7.34}$$

where $\ln p(\mathbf{Y}^k|\mathcal{M}^c) = lpd(\mathbf{Y}^k|\mathcal{M}^c)$ defined in (7.32).

For classification purposes, a good model has to explain the data of its class well but must fail to explain the other classes' data. To express the discriminative capabilities of a model, the metric g_{elpd} is defined. It is the gap (difference) between $\widehat{elpd}$ for the matching model-data pairs (e.g., Trajectory 0 model with Trajectory 0 data) and for the mismatching model-data pairs (e.g., Trajectory 0 model with Trajectory 1 data):

$$g_{elpd} = \widehat{elpd}(\mathcal{Y}^c_v, \mathcal{M}^c) - \widehat{elpd}(\mathcal{Y}^{\neg c}_v, \mathcal{M}^c), \tag{7.35}$$

where $\mathcal{Y}^{\neg c}_v$ represents the dataset of classes other than c, i.e., $\mathcal{Y}^{\neg c}_v$ are the mismatching model-data pairs. The models with higher g_{elpd} have better discriminative performance and are more suitable for the classification task. The results are indicated in Table 7.1.

Another undesired aspect is a high model variance (overfitting), leading to a poor model generalization. To measure the model variance, we compare the performance on the training data with the performance on the validation data. Formally, we define the training-validation *elpd* gap g_{var} as:

$$g_{var} = \widehat{elpd}(\mathcal{Y}^c_{tr}, \mathcal{M}^c) - \widehat{elpd}(\mathcal{Y}^c_v, \mathcal{M}^c). \tag{7.36}$$

The discriminative power and the variance are coupled attributes of the model. They move in opposite directions with the change of the continuity parameter κ. We need a high discriminative power g_{elpd} and a low variance g_{var} at the same time. Therefore, we conducted a series of experiments to find the empiric optimal choice of κ. The effect of the value of κ is dependent on the data, as can be seen in Figures 7.5a, 7.5c, and 7.5b.

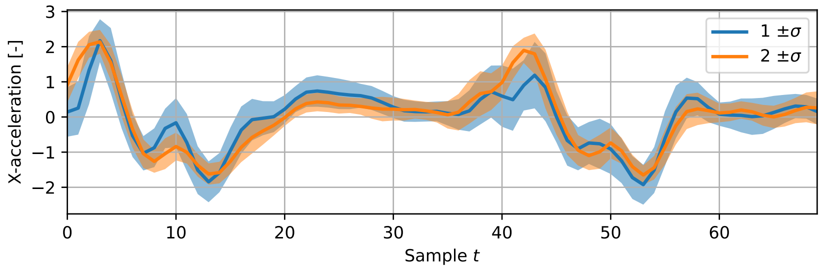
The zero or near-zero values of κ correspond to the classic-trained TV-HGMM. The results show that even small values of κ lower the variance while maintaining the discriminative power. Overall, the application of the continuity preference results in better classification performance.

7.4.2 UCR & UEA Sony AIBO Surface Dataset

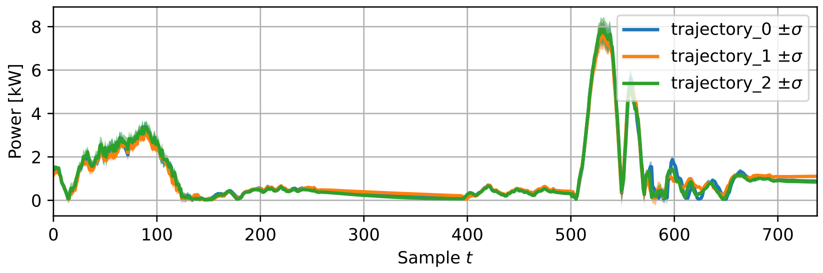
First tests were conducted on a public benchmark dataset from the UEA & UCR dataset repository [126]. Dataset *Sony AIBO Surface1* [127] was used, as it is taken from the internal accelerometer from the quadruped mobile robot Sony AIBO, for which the assumption of the continuous evolution of dynamics parameters is reasonable. In this paper, we refer to this dataset in short as *SonyAIBO*. We took the whole dataset of 621 examples and partitioned it into 6 training and 615 testing examples. The dataset is visualized in Figure 7.4a.

The results of the experiments are in Figure 7.5a. The color encodes the value of $\sqrt{\kappa}$. The darkest point in the top-left corner corresponds to the classic training of the model without the continuity preference. The *SonyAIBO* dataset contains a lot of noise, and the similarity of the examples is much lower than in the case of the industrial robot datasets in the following sections. However, the continuity preference lowers the g_{var} by half while also increasing the discriminative power g_{elpd} twice compared to the time-varying (TV) model. To compare with benchmark methods for the regression task, we selected $\sqrt{\kappa} = 69$ as this value shows the best trade-off between g_{var} and g_{elpd}, i.e., g_{var} as low as possible and g_{elpd} as high as possible at the same time. The values of g_{var}, g_{elpd} and $\sqrt{\kappa}$ are listed in Table 7.1.

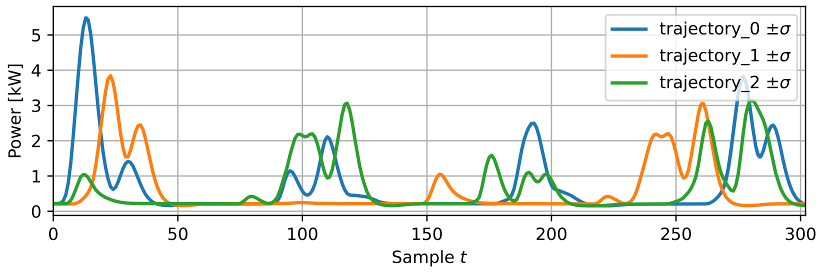
For the classification task, the comparison of the performance is in Table 7.2. The best performing models are the TV-HGMM in its classic variant and in our continuity preferring



(a) Sony AIBO



(b) KR5



(c) KR210

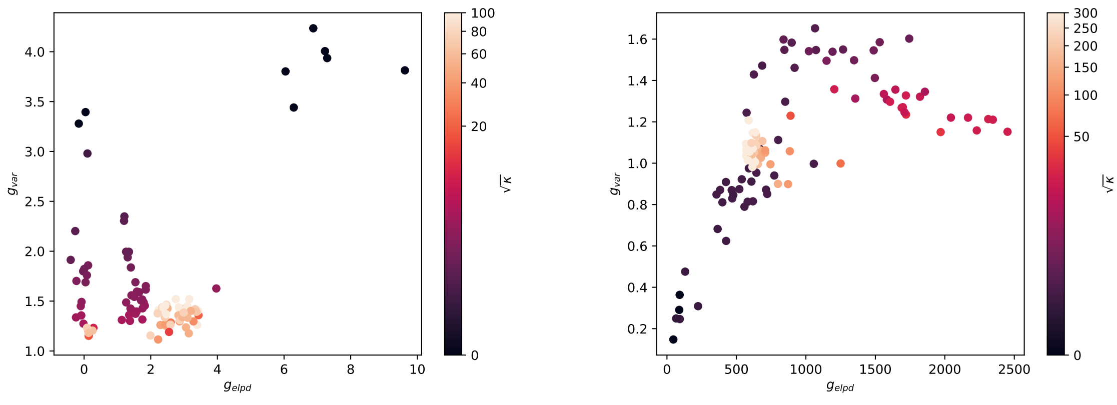
Figure 7.4: Datasets used in experiments. Mean and $\pm 1\sigma$ range of the measured values in the dataset.

variant. The accuracy of our model is 2 % higher than the classic TV-HGMM, while the cross-entropy (CE) remains very similar. The difference between 88 % for TV-HGMM and 90 % accuracy for our approach is significant. It is getting closer to the best achievable accuracy for the *SonyAIBO* dataset, which is reported [127] to be 91.84 % achieved by the Fast Shapelet Forest algorithm that trained on a training dataset that was three times bigger than ours.

7.4.3 KUKA KR5 in Laboratory Conditions

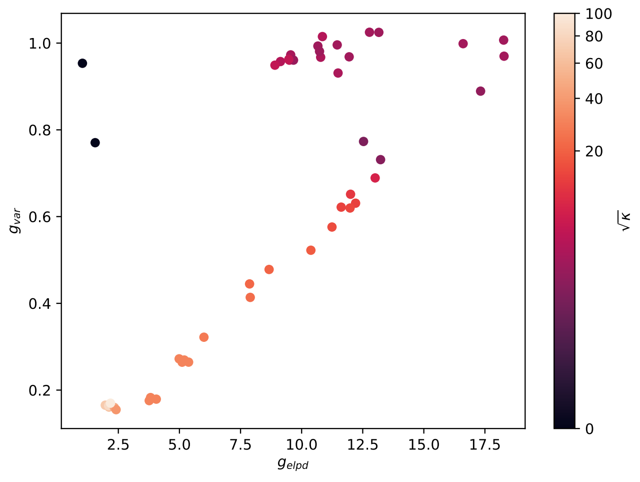
The laboratory experiments on a robotic manipulator KUKA KR5 were conducted, creating dataset *KR5*. In this experiment, the robot trajectories were designed to resemble one another very much to demonstrate the power of the continuity-preference algorithm. Three similar trajectories (classes) that are difficult to distinguish were generated. Each makes the robot move in all six axes on the route through 20 points in Cartesian space. The trajectories differ only a bit in Cartesian points positions, and in Trajectory 2, some points are left out entirely, as depicted exaggeratedly in Figure 7.6. The power consumption of the designed trajectories is our input data. Their shape is depicted in Figure 7.4b. The data corpus consists of 900 sequences, evenly divided into three classes. Six sequences from each class were selected for training. The rest was used for testing.

The Pareto-optimization search is depicted in Figure 7.5c. The performance shows a shift



(a) SonyAIBO

(b) KR210



(c) KR5

Figure 7.5: Regression Pareto experiments with training using different values for continuity parameter κ. g_{var} represents variance, g_{elpd} discriminative power. Values towards the right-bottom of the plane are the best trade-off in performance.

towards the bottom-right corner with $\sqrt{\kappa}$ close to 24. The top-left point represents the classic TV-HGMM. By applying the continuity preference, discriminative power g_{elpd} becomes roughly twice higher and variance g_{var} roughly half than the classically trained TV-HGMM. We selected the value of $\sqrt{\kappa} = 24$ for comparison in Table 7.1.

For the classification task, the achieved accuracy and cross-entropy loss (CE) are listed for comparison in Table 7.2. The time-invariant (TI) version shows poor accuracy as the complexity of such a model is too low to capture the subtle differences that we wish to distinguish. The classic time-varying (TV) model achieves an accuracy of 97 %, one percent less than the continuity preferring model. Still, its cross-entropy loss is almost two times higher than the loss of our continuity preferring model. This is expected according to the observed values of g_{elpd} and g_{var} in Table 7.1. The BOSSE result is nearly perfect, but its computational demands are prohibitive for the method to be used in an online fashion. We keep it for reference that it is possible to achieve nearly perfect classification if online evaluation is not required.

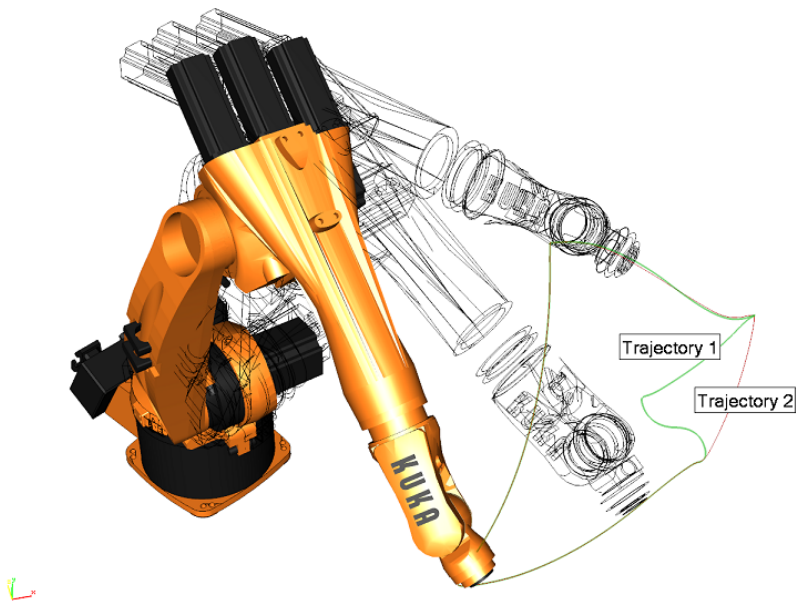


Figure 7.6: An exaggerated example of two trajectories difference in 1 point in Cartesian space.

7.4.4 KUKA KR210 R2700 in Production Line

The data were measured on a live production line in a car body shop in Skoda Auto, which uses the KR210 robots that are of a similar type as the KR5 robot used in Section 7.4.3. We call this dataset *KR210*. The dataset consists of 3 distinct operations depicted in Figure 7.4c. The data corpus consists of 178 sequences divided into a training and a test set. The training examples are stratified by class as 10, 6, and 2 examples, i.e., the minimum of 2 training examples per class is maintained. The rest of the examples was used for the performance evaluation.

The discriminative performance in terms of g_{elpd} and the overfitting in terms of g_{var} are listed in Table 7.1. The classification performance is in Table 7.2. Almost all of the methods can classify this dataset perfectly as the classes are well distinguishable. It is important that TV-HGMM is one of the lowest complexity models that are able to classify this dataset perfectly. Additionally, our approach does not deteriorate the performance and slightly improves the cross-entropy loss (CE). From the regression metrics perspective, our method significantly improves margin g_{elpd} for discriminating between classes, see Table 7.1.

Dataset:	SonyAIBO		KR5		KR210	
Model	$g_{elpd}\uparrow$	$g_{var}\downarrow$	$g_{elpd}\uparrow$	$g_{var}\downarrow$	$g_{elpd}\uparrow$	$g_{var}\downarrow$
TI-HGMM	3.18	1.51	0.21	0.06	43.18	0.72
TV-HGMM	3.26	2.00	4.02	0.50	45.89	0.15
Ours	**3.2**	**1.02**	**6.12**	**0.31**	**871.77**	**0.90**
$\sqrt{\kappa}$	69		24		116	

Table 7.1: Regression performance of the best performing training runs of models for the three selected datasets. Continuity preference was set for respective datasets as $\sqrt{\kappa}_{SonyAIBO} = 69$, $\sqrt{\kappa}_{KR5} = 24$, and $\sqrt{\kappa}_{KR210} = 116$. The arrows in label row indicate whether a bigger ($\uparrow$) or smaller ($\downarrow$) value is desired.

Dataset:	SonyAIBO		KR5		KR210	
Model	Acc. ↑	CE ↓	Acc. ↑	CE ↓	Acc. ↑	CE ↓
KNN	0.74	9.100	0.82	6.16	1.00	0.000
KNN-DTW	0.70	10.330	0.74	8.88	1.00	0.000
TI-HGMM	0.60	1.187	0.29	38.438	0.31	20.324
TV-HGMM	0.88	0.339	0.97	1.011	1.00	0.002
BOSSE	0.82	0.51	0.9988	0.02	1.00	0.000
Ours	**0.90**	**0.333**	**0.98**	**0.519**	**1.00**	**0.000**
$\sqrt{\kappa}$	69		24		116	

Table 7.2: Mean classification performance of the models for the three selected datasets. Continuity preference was set for respective datasets as $\sqrt{\kappa}_{SonyAIBO} = 69$, $\sqrt{\kappa}_{KR5} = 24$, and $\sqrt{\kappa}_{KR210} = 116$. The Acc. stands for accuracy, the CE stands for cross-entropy loss. The arrows in label row indicate whether a bigger (↑) or smaller (↓) value is desired. BOSSE algorithm runs only offline.

7.5 Discussion and Conclusion

The presented novel approach of the continuity preference in training the time-varying hidden Gauss-Markov models (HGMM) balances the advantages of the great explanatory power of the time-varying HGMM with robustness and fast learning rate of time-invariant HGMM. The resulting continuity-preferring EM algorithm can train TV-HGMM on minuscule datasets of only a few examples yielding a model that generalizes outside its training data better than its classically trained version.

The amount of data required for training the TV-HGMM models is enormous. Our approach's significance resides in enabling the TV-HGMM models to be applied in practice on repetitive processes by significantly reducing the data requirements for the training.

The TV-HGMM can boost modeling by spreading the time-varying parameters across the time dimension. In turn, a much lower number of states is required resulting in more stable learning performance. Each time-step is treated independently in the classical EM algorithm with no temporal co-dependence of parameter transitions (continuity preference). Consequently, each time-step can converge after a different number of EM iterations. This effect leads to a model that has some parts under-fitted and some parts over-fitted. Our approach suppresses such an undesired behavior, resulting in a better numerical stability during the training and online classification.

By introducing the preference of continuity in transition matrices, we amplified the direct influence of the neighboring time-steps. It is not just a variant of the standard Tikhonov regularization. We are not pulling the parameters towards some particular value but instead pulling parameters closer to each other while maintaining their freedom to drift as a whole. The resulting solution is biased towards a model, which is smooth in its parameters.

Our approach increases the discriminative power of the trained models, as demonstrated through experiments on real robots that exhibit the continuity property. Although the primary use case was the industrial six-axis robot [124], there are many kinds of other machines that exhibit the continuity property too. We demonstrated the generalization of our approach on the UEA & UCR public dataset *Sony AIBO Surface 1* [127].

The ability to tightly model multivariate signals also opens other possible applications of such models. For example, the deviation detection task requires a model as tight as possible, and the training examples of the deviations are available in only a few instances. Such a task can be applied, e.g., in fault detection, monitoring of conditions deterioration, or predictive maintenance. The continuity-preference method learns more accurate TV-HGMM parameters from less data, which allows the TV-HGMM to be deployed in practical applications.

The software is available on GitHub [128], the datasets created for this paper are available online at [124].

Chapter 8

Final Conclusions

This dissertation goals require to cover multiple domains of computer science, namely machine learning, control theory, signal processing, and statistical modeling. Each chapter of the dissertation describes the best fitting approach to its cross-functional requirements. Moreover, each chapter contributed with something new to the studied problematics, which is demonstrated by published works [3], [10], [11], [22], [91], [98] and one more submitted work about motif discovery.

We started with a bottom-up analysis of structured model of a 6-DOF robot dynamics which was used as a test case of this work. The basis of the expected structure of the models was developed in Chapter 2. We have shown, that due to a high complexity of such a model, it is advantageous to apply techniques of machine learning to identify the parameters of the structured model instead of a first-principle-based identification. Machine learning techniques prove to bring better scalability while keeping high accuracy of identified parameters.

In Chapter 3, we investigated the manual segmentation of time series. The simple case of well distinguishable operations that can be segmented by signal plateaus is hardly guaranteed in practice. Pattern recognition on the other hand is a robust approach that can be applied on various kinds of time series. For that reason, we decided to follow the path of distinguishing operations using pattern recognition. This approach assumed that the pattern is manually selected and we need to automatically locate it within the time series. We focused on two cases. First, the brute force sliding window approach, and second, the feature-extraction preprocessing method which is suitable when computational resources are limited. The feature extraction based on peak detection proves to be well suited for 6-DOF robots as experimentally verified. The time series classification accuracy was in range of $0.7 - 0.9$, but a lot of manual tuning for detection was required, which is slow, expensive and prone to errors.

Thus, in Chapter 4 we investigated the topic of motif discovery – a problem that we showed to be adaptable for segmenting the time series. We augmented the formulation of a typical motif discovery task. We also presented a data processing pipeline that builds a loosely structured pattern collection – an automatically collected basis for a training data set.

Next, we shifted our focus to structuring the pattern collection into a proper training data set. That brings a requirement of an unsupervised machine learning technique – clustering of time series segments by their pair-wise similarity. Chapter 5 discusses the topic of clustering and proposes a similarity measure that considers the information content of the samples and compares unequal-length sequences. The best suited clustering algorithm is OPTICS, a type of an agglomerative clustering that can discover even very irregular clusters in the high-dimensional space of time-series subsequences. The time complexity of the similarity measure is quite demanding, but it can be parallelized, which improves its performance significantly. This presents an advantage compared to the popular DTW measure, which cannot be parallelized.

Our next step was to investigate other method of pattern modeling that takes into account the known system structure and models its behavior stochastically. In Chapter 6, we explored the HMM pattern modeling approach, which is a well established model for time series. We laid down the basis for probabilistic analysis of structured HMM model of a 6-DOF robot and we ran a preliminary proof-of-concept experiments on a single axis of a robot. In the experiments, we verified that our proposed modeling methods are suitable for a sequential classification of time series. This means that methods of HMM can be used for an online segmentation of time series based on a set of known patterns. At the same time, we found out that the method requires a lot of training data to classify the sequences reliably.

In Chapter 7, we enhanced the results from 6 by reducing the amount of training data required by TV-HGMM. The main problem in training the TV-HGMM is the high variance in the time-neighboring parameters. We interpret this observation in terms of machine learning as a high variance of model, i.e., overfitting, and we introduced an advanced regularization technique denoted as continuity preference. This technique enforces the time-neighboring parameters of TV-HGMM to differ only a little. We also derived theoretical optimization formulas resulting in an elegant augmentation of a standard EM learning algorithm. The continuity preference significantly improved the classification performance of the model when only few training examples are available. Extensive experiments on real-life data sets have shown the classification accuracy to be above 0.98.

8.1 Main Contributions

We summarize the main contributions that were published in the author's original work in [3], [10], [11], [22], [91], [98]. To explicitly spell them out in the following list:

- **Collision table strategy for accelerating motif discovery algorithms.** We introduced a novel view of collision table in motif discovery enabling it to be modularly applied to boost performance of a wide class of motif discovery algorithms. We analyzed the computational complexity reduction impact of the method and presented it on real life data. For details, see Chapter 4. The results were submitted for publishing.

- **Information weighted similarity measure for unequal length sequences.** We proposed a novel parallelizable similarity measure that yields easily interpretable results. It is a binary symmetric operation that highlights samples of high estimated information and lowers weight of low information samples. This measure allows us to cluster sequences of different lengths. Details are provided in Chapter 5. Results were published in [98].

- **Time series feature extraction effective for 6-DOF robot power consumption data.** We proposed a particularly effective feature extraction method for 6-DOF robot power consumption data which shows very convincing detection rates while reducing the computational load significantly. Details are described in Chapter 3. The results were published in [10], [11].

- **HMM based modeling of 6-DOF robots with hidden control inputs.** We showed how HMM can be applied on industrial robot provided only the power consumption data are available and the control inputs of the robot are hidden. This contribution enables a non-intrusive modeling of robots that are already deployed in production without interrupting the manufacturing process. This property allows for easier deployment of the proposed methods to industry. For details, see Chapter 6. The results were published in [3].

- **Continuity preference for few-shot learning of TV-HGMM.** We introduced an assumption of parameter continuity for TV-HGMM and demonstrated how this assumption improves the generalization performance of models trained on data sets counting less than six training examples. By theoretical analysis, we derived a minimalistic augmentation of

EM learning algorithm that is typically used to train TV-HGMM making it easily adoptable for current software frameworks. Details are provided in Chapter 7 and published in [91].

- **Integrated framework for pattern detection in time series – MOD tool.** All the parts presented in this dissertation were integrated into the MOD tool – an integrated pattern detection application that can be deployed to a wide range of domains in industry, where repetitive processes play a key role in production. The MOD tool components are researched under the European research project BRAINE [129]. The MOD tool has been tested at real-life production at industrial use case of batch-heating ovens. Dissemination of the framework, e.g, at the Summer School on Cyber-Physical Systems and Internet-of-Things [130].

8.2 Fulfillment of Dissertation Goals

Here, all the dissertation goals as defined in Section 1.1 are stated again and for each of them an explanation is provided to show how the goal was fulfilled.

- **Design methods for automatic search of apriori-unknown patterns in time series.** We formulated and solved the problem of finding unknown patterns as Motif Discovery problem in Chapter 4. The solution was tested both on synthetic benchmark data sets and on real-life data sets obtained from industrial use cases. Then in Chapter 3, the discovered motifs were searched for in time series and, based on that, the data were segmented. The segmentation was evaluated on energy-consumption data from an industrial robotic cell running in Škoda Auto plant. The results were published in [10] and submitted as [22].

- **Propose a solution to cluster the variable-length sequences to reduce the number of the unsorted patterns into groups containing only patterns which are similar.** Solution to this task is presented in Chapter 5, where we proposed an original similarity measure utilizing the information content of the samples. This similarity measure was used with clustering algorithm OPTICS to automatically determine the number of clusters in the time-series data. The performance was tested on the data from the robotic cell in Škoda Auto. The results were published in [98].

- **Develop machine-learning methods for the identification of the model parameters.** We investigated the HMM stochastic models in Chapter 6 as a means for time-series classification. We showed the time-varying HMM provide good prediction accuracy and are suitable for the time-series classification task. We published our findings in [3]. Further focus on the time-varying hidden Gauss-Markov models TV-HGMM led us to inventing the parameter-continuity-preference method for learning accurate model parameters from very small training data sets as described in Chapter 7. We published this method in [91].

- **Apply the invented methods for an online detection of repetitive behavior observed in time series.** The detection of repetitive behavior was formulated as an iterated time-series-classification task (TSC) and it was used to demonstrate the performance of the entire developed framework in Chapter 7. The findings were published in [91].

- **Validate the results on real-life use cases from industrial robotics.** The modular nature of this dissertation allows validating the results module by module. The individual modules correspond to the individual chapters of the dissertation. Each module was tested on the use cases from industrial robotics. Chapter 7 and our paper [91] present the extensive validation experiments of the integrated solution using the real-life data sets measured on industrial robots.

We methodically proceeded towards the defined goals, we developed the Motif Discovery and Detection Framework and we successfully demonstrated how it automates the task of repetitive-behavior discovery, learning and detection. The efforts resulted in a software application that was deployed in industrial batch-heating ovens. Thus, all the goals of this dissertation were successfully achieved.

8.3 Future Work

To further scale up the proposed methods, future work should focus on accelerating the computations by parallelizing the presented algorithms. The emerging quantum computing deserves special attention as it shows promising directions for accelerating the statistical inference utilized in this dissertation.

The approximation motif discovery algorithms should be further improved especially with respect to their speed to allow for deployment in online applications.

We demonstrated how the parameter-continuity preference in TV-HGMM improves the learning from small data sets. This contribution addresses the practical requirement to provide only a minimal sufficient data set for machine learning tasks. As the continuity in parameters proved to be beneficial for TV-HGMM classification performance, other methods yielding smoother parameters should be investigated. A close relative of HGMM are the Gaussian processes (GP) which do not require the Markov property assumption and exhibit interesting smoothing properties. However, the GP has the time complexity of $\mathcal{O}(N^3)$ for predictions. Future work should investigate GP approximations that show time complexity of $\mathcal{O}(N)$.

All the presented methods are integrated in the Motif Discovery tool developed within the European research project BRAINE [129]. The final goal is to use the stochastic models to automatically detect deviations in the modeled behavior of machines.

Appendix A

Continuous State Hidden Gauss-Markov Models

A.1 Theorems and Useful Expressions

Theorem 1 (Jensen's inequality) *Let $f(x)$ is a concave function of a real-valued random variable $X : \Omega \to \mathbb{R}$ then,*

$$f(\mathbb{E}\{X\}) \geq \mathbb{E}\{f(X)\}.$$

For proof, see, e.g., [52].

Lemma about a linear transformation of normally distributed random vector found, e.g., in [131] are used in following derivations.

Lemma 1 *Let $\mathbf{x} \in \mathbb{R}^n$ is a realization of a normally distributed real-valued random vector $\mathbf{X} : \Omega \to \mathbb{R}^n$, i.e., $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{P})$, with mean value and covariance $\boldsymbol{\mu}, \mathbf{P}$ respectively, and let matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ then,*

$$\mathbf{Ax} \sim \mathcal{N}(\mathbf{A}\boldsymbol{\mu}, \mathbf{APA}^\top),$$

additionally, if a random vector $\mathbf{v} \sim \mathcal{N}(\boldsymbol{\nu}, \mathbf{Q})$ is independent of $\mathbf{x}$, i.e., $\mathbf{x} \perp\!\!\!\perp \mathbf{v}$, then,

$$(\mathbf{Ax} + \mathbf{v}) \sim \mathcal{N}(\mathbf{A}\boldsymbol{\mu} + \boldsymbol{\nu}, \mathbf{APA}^\top + \mathbf{Q}).$$

For proof, see, e.g., [131].

For conditioning joint normal distributions, we need the following lemma from [5].

Lemma 2 (Gaussian refactorization lemma) *Given the vectors $\mathbf{x}, \boldsymbol{\mu} \in \mathbb{R}^q$, the symmetric positive definite (PD) matrix $\mathbf{P} \in \mathbb{R}^{q \times q}$, the vector $\mathbf{y} \in \mathbb{R}^p$, the symmetric PD matrix $\mathbf{S} \in \mathbb{R}^{p \times p}$, and the matrix $\mathbf{F} \in \mathbb{R}^{p \times q}$, then*

$$\mathcal{N}(\mathbf{y}; \mathbf{Fx}, \mathbf{S})\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \mathbf{P}) = \mathcal{N}(\mathbf{y}; \boldsymbol{\omega}, \boldsymbol{\Omega})\mathcal{N}(\mathbf{x}; \boldsymbol{\lambda}, \boldsymbol{\Lambda}), \tag{A.1}$$

where the variables on the right-hand side are defined by

$$\begin{aligned}
\boldsymbol{\Omega} &= \mathbf{S} + \mathbf{FPF}^\top &\text{(A.2)}\\
\boldsymbol{\omega} &= \mathbf{F}\boldsymbol{\mu} &\text{(A.3)}\\
\mathbf{H} &= \mathbf{PF}^\top\boldsymbol{\Omega}^{-1} &\text{(A.4)}\\
\boldsymbol{\Lambda} &= (\mathbf{I} - \mathbf{HF})\mathbf{P} &\text{(A.5)}\\
\boldsymbol{\lambda} &= (\mathbf{I} - \mathbf{HF})\mathbf{P}\boldsymbol{\mu} + \mathbf{Hy}. &\text{(A.6)}
\end{aligned}$$

Remark 1 *When allowing for degenerative case of Gaussian distribution, the covariance matrix can be positive semi-definite (PSD). In case of PSD matrix Ω, the calculation of* (A.5) *becomes*

$$\mathbf{H} = \mathbf{PF}^\top\boldsymbol{\Omega}^\dagger, \tag{A.7}$$

where $\Omega^\dagger$ denotes the pseudo-inverse ($\boldsymbol{\Omega}\boldsymbol{\Omega}^\dagger\boldsymbol{\Omega} = \boldsymbol{\Omega}$).

For proof on both see [5].

Note that the equation (A.1) is Gaussian form of Bayes rule $p(\mathbf{y}|\mathbf{x})p(\mathbf{x}) = p(\mathbf{y})p(\mathbf{x}|\mathbf{y})$. This lemma is used later in this paper for derivation of E-step to express conditioning on observation.

A.2 Detailed Derivation of EM Algorithm

A.2.1 E-step

The E-step calculates the mean values and covariances of the conditional density of the hidden states based on the current data $\mathcal{Y}$ and parameters $\mathbf{\Theta}^i$. Formally:

$$\mathbf{x}_t \sim \mathcal{N}(\mathbf{x}_t; \boldsymbol{\mu}_{t|t}, \mathbf{P}_{t|t}) \tag{A.8}$$

$$\hat{\mathbf{x}}_t = \mathbb{E}_{\mathbf{x}_t|\mathcal{Y};\mathbf{\Theta}^i}\{\mathbf{x}_t\} = \boldsymbol{\mu}_{t|t}. \tag{A.9}$$

This conditional density is possible to be computed either based on forward or backward recursion. Both of these recursions combined give the most accurate result as both the information accumulated from the future samples and the information accumulated from the past samples are merged together. To compute the backward recursion, a sequence of the future samples must be available to evaluate the likelihood density for current hidden state vector $\mathbf{x}_t$. This requires the entire example sequence of training data to be available during the training process.

For the sake of clarity, we omit the upper indexing in this section, keeping in mind that all measured sequences share the same set of parameters $\boldsymbol{\mu}_1$, $\mathbf{P}_1$, $\mathbf{A}_t$, $\mathbf{B}$, $\mathbf{Q}_t$ and $\mathbf{R}$. The forward recursion algorithm estimates mean and covariance of density $\mathcal{N}(\mathbf{x}_t; \boldsymbol{\mu}_{t|t}, \mathbf{P}_{t|t})$ for each time instance. The algorithm consists of two steps – time update and data update step. First, the initial values are set for time $t = 1$ acting as time update as

$$\boldsymbol{\mu}_{1|0} = \boldsymbol{\mu}_1 \tag{A.10}$$

$$\mathbf{P}_{1|0} = \mathbf{P}_1. \tag{A.11}$$

Then the data update is calculated for $t = 1$ based on Equations (A.14)–(A.17) that follow. Alternating between the time and the data update step the parameters of the conditional densities are calculated for each time instance t. The time update step

$$\mathbf{P}_{t|t-1} = \mathbf{Q}_t + \mathbf{A}_t\mathbf{P}_{t-1|t-1}\mathbf{A}_t^\top, \tag{A.12}$$

$$\boldsymbol{\mu}_{t|t-1} = \mathbf{A}_t\boldsymbol{\mu}_{t-1|t-1}, \tag{A.13}$$

and the data update step is computed as

$$\mathbf{\Sigma}_t = \mathbf{R} + \mathbf{B}\mathbf{P}_{t|t-1}\mathbf{B}^\top, \tag{A.14}$$

$$\mathbf{G}_t = \mathbf{P}_{t|t-1}\mathbf{B}^\top\mathbf{\Sigma}_t^{-1}, \tag{A.15}$$

$$\mathbf{P}_{t|t} = (\mathbf{I} - \mathbf{G}_t\mathbf{B})\mathbf{P}_{t|t-1}, \tag{A.16}$$

$$\boldsymbol{\mu}_{t|t} = (\mathbf{I} - \mathbf{G}_t\mathbf{B})\boldsymbol{\mu}_{t|t-1} + \mathbf{G}_t\mathbf{y}_t. \tag{A.17}$$

$\boldsymbol{\mu}_{t|t}$ is the best estimate of the hidden state $\hat{\mathbf{x}}_t$ using only the forward recursion with the past data up to time t. Finally, the statistics $\mathbf{H}_t$ for cross-covariance of the time-adjacent states is calculated at this point as

$$\mathbf{H}_t = \mathbf{P}_{t-1|t-1}\mathbf{A}_t^\top\mathbf{P}_{t|t-1}^{-1}. \tag{A.18}$$

(A.18) is used later in the M-step of the EM algorithm.

The Equations (A.12)–(A.17) represent the standard Kalman filtering principle in a slightly rearranged manner adopted from [5]. By applying the algebraic association rule, we get the exact form of Kalman filtering equations as shown, e.g., in [132].

A.2.2 M-step

In the M-phase, the EM algorithm forgets the parameters $\boldsymbol{\mu}_1$, $\mathbf{P}_1$, $\mathbf{A}_t$, $\mathbf{B}$, $\mathbf{Q}_t$ and $\mathbf{R}$ used in the preceding E-step and estimates new values for them based on the original measurement example sequences $\mathcal{Y}$ and their corresponding annotations $\mathcal{X}$.

For the derivation of maximization we will need following lemma from [[133], Ch. 10].

Lemma 3 *Let* $\mathbf{x}$ *be normally distributed random vector, then*

$$\begin{aligned}\int \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \mathbf{P})(\mathbf{x}^\top \mathbf{F}\mathbf{x} + \mathbf{x}^\top \mathbf{f} + f_0)d\mathbf{x} = \\ = \mathrm{tr}(\mathbf{F}(\mathbf{P} + \boldsymbol{\mu}\boldsymbol{\mu}^\top)) + \boldsymbol{\mu}^\top \mathbf{f} + f_0. \qquad \text{(A.19)}\end{aligned}$$

Formally, the i-th M-step iteration of the algorithm maximizes the lower bound (7.6) by solving

$$\boldsymbol{\Theta}^{i+1} = \arg\max_{\boldsymbol{\Theta}} Q(\boldsymbol{\Theta}, \boldsymbol{\Theta}^i), \qquad \text{(A.20)}$$

$$Q(\boldsymbol{\Theta}, \boldsymbol{\Theta}^i) = \mathbb{E}_{\mathcal{X}|\mathcal{Y};\boldsymbol{\Theta}^i}\left\{\ln p(\mathcal{Y}, \mathcal{X}; \boldsymbol{\Theta})\right\}. \qquad \text{(A.21)}$$

This auxiliary function $Q(\boldsymbol{\Theta}, \boldsymbol{\Theta}^i)$ is decomposed into three sub-functions and optimized separately as follows.

$$\begin{aligned}
Q(\boldsymbol{\Theta}, \boldsymbol{\Theta}^i) &= Q_1 + Q_X + Q_Y, && \text{(A.22)}\\
Q_1 &= \sum_{k=1}^{K} \int p(\mathbf{x}_1|\mathbf{Y}^k_{T_k}) \ln p(\mathbf{x}^k_1) d\mathbf{x}^k_1, && \text{(A.23)}\\
Q_X &= \sum_{t=2}^{N_{max}} \sum_{k=1}^{K_t} \iint p(\mathbf{x}^k_t, \mathbf{x}^k_{t-1}|\mathbf{Y}^{N_k}_1) \\
&\quad \times \ln p(\mathbf{x}^k_t|\mathbf{x}^k_{t-1}) d\mathbf{x}^k_{t-1} d\mathbf{x}^k_t \\
&= \sum_{t=2}^{N_{max}} \sum_{k=1}^{K_t} I_{tk}, && \text{(A.24)}\\
Q_Y &= \sum_{t=1}^{N_{max}} \sum_{k=1}^{K_t} \int p(\mathbf{x}^k_t|\mathbf{Y}^k_{T_k}) \\
&\quad \times \ln p(\mathbf{y}^k_t|\mathbf{x}^k_t) d\mathbf{x}^k_t. && \text{(A.25)}
\end{aligned}$$

By substituting Gaussian distributions in place of pdfs, and using lemma (3), we obtain expressions for each Equation (A.23)-(A.25).

$$\begin{aligned}
Q_1 &= \sum_{k=1}^{K} \int \mathcal{N}(\mathbf{x}^k_1; \boldsymbol{\mu}^k_{1|T_k}, \mathbf{P}^k_{1|T_k}) \ln \mathcal{N}(\mathbf{x}^k_1; \boldsymbol{\mu}_1, \mathbf{P}_1) && \text{(A.26)}\\
&= -\frac{KL}{2}\ln(2\pi) + \frac{K}{2}\ln|\mathbf{P}_1|^{-1} \\
&\quad -\frac{1}{2}\sum_{k=1}^{K} \mathrm{tr}\left\{\mathbf{P}_1^{-1}\left(\mathbf{P}^k_{1|T_k} + \boldsymbol{\varepsilon}^k_1 \boldsymbol{\varepsilon}^{k\top}_1\right)\right\}, && \text{(A.27)}
\end{aligned}$$

where L is the state-space model dimension and $\boldsymbol{\varepsilon}^k_1 = \boldsymbol{\mu}^k_{1|T_k} - \boldsymbol{\mu}_1$.

For the component Q_X the integral (i.e., the expectation) is evaluated as:

$$I_{tk} = -\frac{1}{2}\left\{L\ln(2\pi) - \ln|\mathbf{Q}_t^{-1}| + I^{(1)}_{tk} - I^{(2)}_{tk} + I^{(3)}_{tk}\right\},$$

where

$$\begin{aligned}
I_{tk}^{(1)} &= \int \mathcal{N}(\mathbf{x}_t^k; \boldsymbol{\mu}_{t|T_k}^k, \mathbf{P}_{t|T_k}^k) \mathbf{x}_t^{k\top} \mathbf{Q}_t^{-1} \mathbf{x}_t^k d\mathbf{x}_t^k \\
&= \mathrm{tr}(\mathbf{Q}_t^{-1} \mathbf{C}_{\mathbf{x}_t\mathbf{x}_t}^k) \\
I_{tk}^{(2)} &= \iint \mathcal{N}\left(\mathbf{x}_{t,t-1}^k; \boldsymbol{\mu}_{t,t-1|T_k}^k, \mathbf{P}_{t,t-1|T_k}^k\right) \\
&\quad \times \left\{\mathbf{x}_t^{k\top} \mathbf{Q}_t^{-1} \mathbf{A}_t \mathbf{x}_{t-1}^k + \mathbf{x}_{t-1}^{k}{}^{\top} \mathbf{A}_t^\top \mathbf{Q}_t^{-1} \mathbf{x}_t^k\right\} d\mathbf{x}_{t-1}^k d\mathbf{x}_t^k \\
&= 2\,\mathrm{tr}\left(\mathbf{Q_t^{-1} A_t} \mathbf{C}_{\mathbf{x}_t\mathbf{x}_{t-1}}^k\right) \\
I_{tk}^{(3)} &= \int \mathcal{N}(\mathbf{x}_{t-1}^k; \boldsymbol{\mu}_{t-1|T_k}^k, \mathbf{P}_{t-1|T_k}^k) \mathbf{x}_{t-1}^{k\top} \mathbf{A}_t^\top \mathbf{Q}_t^{-1} \mathbf{A}_t \mathbf{x}_{t-1}^k d\mathbf{x}_{t-1}^k \\
&= \mathrm{tr}\left(\mathbf{A_t^\top Q_t^{-1} A_t} \mathbf{C}_{\mathbf{x}_{t-1}\mathbf{x}_{t-1}}^k\right),
\end{aligned}$$

where we used the cyclic property of the trace operator and the substitution of

$$\begin{aligned}
\mathbf{C}_{\mathbf{x}_t\mathbf{x}_t}^k &= \mathbf{P}_{t|T_k} + \boldsymbol{\mu}_{t|T_k}^k \boldsymbol{\mu}_{t|T_k}^{k\top}, \\
\mathbf{C}_{\mathbf{x}_t\mathbf{x}_t} &= \frac{1}{K_t} \sum_{k=1}^{K_t} \mathbf{C}_{\mathbf{x}_t\mathbf{x}_t}^k, && \text{(A.28)} \\
\mathbf{C}_{\mathbf{x}_t\mathbf{x}_{t-1}}^k &= \mathbf{P}_{t,t-1|T_k} + \boldsymbol{\mu}_{t|T_k}^k \boldsymbol{\mu}_{t-1|T_k}^{k\top} \\
\mathbf{P}_{t,t-1|T_k} &= \mathbf{P}_{t|T_k} \mathbf{H}_t \\
\mathbf{C}_{\mathbf{x}_t\mathbf{x}_{t-1}} &= \frac{1}{K_t} \sum_{k=1}^{K_t} \mathbf{C}_{\mathbf{x}_t\mathbf{x}_{t-1}}^k. && \text{(A.29)}
\end{aligned}$$

Putting all the expressions together, the term Q_X can be expressed as

$$\begin{aligned}
Q_X &= -\frac{1}{2} \sum_{t=2}^{T_{max}} \sum_{k=1}^{K_t} \Big\{ L \ln(2\pi) - \ln |\mathbf{Q}_t^{-1}| \\
&\quad + \mathrm{tr}(\mathbf{Q}_t^{-1} \mathbf{C}_{\mathbf{x}_t\mathbf{x}_t}^k) \\
&\quad - 2\,\mathrm{tr}\left(\mathbf{Q_t^{-1} A_t} \mathbf{C}_{\mathbf{x}_t\mathbf{x}_{t-1}}^k\right) \\
&\quad + \mathrm{tr}\left(\mathbf{A_t^\top Q_t^{-1} A_t} \mathbf{C}_{\mathbf{x}_{t-1}\mathbf{x}_{t-1}}^k\right) \Big\}. && \text{(A.30)}
\end{aligned}$$

As for the term Q_Y, the procedure is similar.

$$\begin{aligned}
Q_Y &= -\frac{1}{2} \sum_{t=2}^{N_{max}} \sum_{k=1}^{K_t} \left\{ L \ln(2\pi) - \ln |\mathbf{R}_t^{-1}| + \mathcal{I}_{tk} \right\}, && \text{(A.31)} \\
\mathcal{I}_{tk} &= \int \mathcal{N}(\mathbf{x}_t^k; \boldsymbol{\mu}_{t|T_k}^k, \mathbf{P}_{t|T_k}) \ln \mathcal{N}(\mathbf{y}_t^k; \mathbf{B}_t \mathbf{x}_t^k, \mathbf{R}_t) d\mathbf{x}_t^k \\
&= \int \mathcal{N}(\mathbf{x}_t^k; \boldsymbol{\mu}_{t|T_k}^k, \mathbf{P}_{t|T_k}) \\
&\quad \times \{\mathbf{y}_t^{k\top} \mathbf{R}_t^{-1} \mathbf{y}_t^k - \mathbf{x}_t^{k\top} \mathbf{B}_t^\top \mathbf{R}_t^{-1} \mathbf{y}_t^k \\
&\quad - \mathbf{y}_t^{k\top} \mathbf{R}_t^{-1} \mathbf{B}_t \mathbf{x}_t^k + \mathbf{x}_t^{k\top} \mathbf{B}_t^\top \mathbf{R}_t^{-1} \mathbf{B}_t \mathbf{x}_t^k \} d\mathbf{x}_t^k && \text{(A.32)} \\
&= \mathrm{tr}(\mathbf{R}_t^{-1} \mathbf{C}_{\mathbf{y}_t\mathbf{y}_t} - 2\mathbf{R}_t^{-1} \mathbf{C}_{\mathbf{y}_t\mathbf{x}_t} \mathbf{B}_t^\top) \\
&\quad + \mathrm{tr}(\mathbf{B}_t^\top \mathbf{R}_t^{-1} \mathbf{B}_t \mathbf{C}_{\mathbf{x}_t\mathbf{x}_t}), && \text{(A.33)}
\end{aligned}$$

where we applied substitution by correlation matrices of the form

$$\mathbf{C}_{\mathbf{y}_t\mathbf{y}_t} = \frac{1}{K_t}\sum_{k=1}^{K_t}\mathbf{y}_t^k\mathbf{y}_t^{k^\top}, \quad (A.34)$$

$$\mathbf{C}_{\mathbf{y}_t\mathbf{x}_t} = \frac{1}{K_t}\sum_{k=1}^{K_t}\mathbf{y}_t^k\boldsymbol{\mu}_{t|T_k}^{k^\top}. \quad (A.35)$$

Note that the Equations (7.14), (7.15), (A.34), and (A.35) assume the mean value to be exact and known at the time of calculations. In practice, the mean is estimated as a sample mean, so the correlation (covariance) matrices should be normalized by the factor $1/(K_t - 1)$ instead of $1/K_t$ to obtain an unbiased estimate. This is called Bessel's correction [134]. We keep the theoretical form here for readability.

Since time-invariant $\mathbf{B}$ and, $\mathbf{R}$ are calculated, time indices t in (A.31)-(A.33) can be dropped and the correlation matrices are averaged as

$$\mathbf{C}_{\mathbf{yy}} = \frac{1}{T_{max}}\sum_{t=1}^{T_{max}}\mathbf{C}_{\mathbf{y}_t\mathbf{y}_t}, \quad (A.36)$$

$$\mathbf{C}_{\mathbf{yx}} = \frac{1}{T_{max}}\sum_{t=1}^{T_{max}}\mathbf{C}_{\mathbf{y}_t\mathbf{x}_t}, \quad (A.37)$$

$$\mathbf{C}_{\mathbf{xx}} = \frac{1}{T_{max}}\sum_{t=1}^{T_{max}}\mathbf{C}_{\mathbf{x}_t\mathbf{x}_t}. \quad (A.38)$$

To get optimal estimates of parameters $\mathbf{A}_t, \mathbf{B}, \mathbf{Q}_t, \mathbf{R}, \boldsymbol{\mu}_1, \mathbf{P}_1$, the optimization task from (A.21) is ready to be solved by parts now. Solve $\mathcal{D}_{\mathbf{A}_t}\{Q_X\} = 0$ for $\mathbf{A}_t$, $\mathcal{D}_{\mathbf{Q}_t}\{Q_X\} = 0$ for $\mathbf{Q}_t$, $\mathcal{D}_{\mathbf{B}}\{Q_Y\} = 0$ for $\mathbf{B}$, and $\mathcal{D}_{\mathbf{R}}\{Q_Y\} = 0$ for $\mathbf{R}$. For that, standard matrix calculus is used, and additionally, we point out the identity:

$$\mathcal{D}_{\mathbf{F}}\{\mathrm{tr}(\mathbf{F}^\top\mathbf{S}_1\mathbf{F}\mathbf{S}_2)\} = \mathbf{S}_1\mathbf{F}\mathbf{S}_2 + \mathbf{S}_1^\top\mathbf{F}\mathbf{S}_2^\top. \quad (A.39)$$

Solving these equations we obtain following optimums.

$$\mathbf{A}_t^{i+1} = \mathbf{C}_{\mathbf{x}_t\mathbf{x}_{t-1}}\mathbf{C}_{\mathbf{x}_{t-1}\mathbf{x}_{t-1}}^{-1}, \quad (A.40)$$

$$\mathbf{Q}_t^{i+1} = \mathbf{C}_{\mathbf{x}_t\mathbf{x}_t} - \mathbf{C}_{\mathbf{x}_t\mathbf{x}_{t-1}}\mathbf{C}_{\mathbf{x}_{t-1}\mathbf{x}_{t-1}}^{-1}\mathbf{C}_{\mathbf{x}_t\mathbf{x}_{t-1}}^\top, \quad (A.41)$$

$$\mathbf{B}^{i+1} = \mathbf{C}_{\mathbf{yx}}\mathbf{C}_{\mathbf{xx}}^{-1}, \quad (A.42)$$

$$\mathbf{R}^{i+1} = \mathbf{C}_{\mathbf{yy}} - \mathbf{C}_{\mathbf{yx}}\mathbf{C}_{\mathbf{xx}}^{-1}\mathbf{C}_{\mathbf{yx}}^\top. \quad (A.43)$$

For the parameters $\boldsymbol{\mu}_1, \mathbf{P}_1$, solve $\mathcal{D}_{\boldsymbol{\mu}_1}\{Q_1\} = 0$ for $\boldsymbol{\mu}_1$ yielding

$$\boldsymbol{\mu}_1^{i+1} = \frac{1}{K}\sum_{k=1}^{K_t}\boldsymbol{\mu}_{1|T_k}^k \quad (A.44)$$

Defining $\boldsymbol{\varepsilon}_1^{k,i+1} = \boldsymbol{\mu}_{1|T_k}^k - \boldsymbol{\mu}_1^{i+1}$ and then solving $\mathcal{D}_{\mathbf{P}_1^{-1}}\{Q_1\} = 0$ for $\mathbf{P}_1$ yields

$$\mathbf{P}_1^{i+1} = \frac{1}{K_1}\sum_{k=1}^{K_1}\{\mathbf{P}_{1|T_k} + \boldsymbol{\varepsilon}_1^{k,i+1}\boldsymbol{\varepsilon}_1^{k,i+1^\top}\}. \quad (A.45)$$

These results are then used for parameters update in the $(i+1)$-th M-step.

It is worth mentioning that the autocorrelation matrices $\mathbf{C}_{\mathbf{y}_t\mathbf{y}_t}$, $\mathbf{C}_{\mathbf{yy}}$, $\mathbf{C}_{\mathbf{x}_t\mathbf{x}_t}$, $\mathbf{C}_{\mathbf{yy}}$ and also the covariances $\mathbf{Q}_t$ and $\mathbf{R}$ are all symmetric positive semi-definite matrices. In other words, all its eigenvalues are real-valued non-negative numbers. This condition may not be met due to the numerical realization of the algorithm. In such a case, some kind of regularization is necessary. In our experimental setup, we used eigenvalue regularization, replacing the negative eigenvalues with small positive values. This procedure can be viewed as artificially induced uncertainty in the corresponding random variable.

List of Candidate's Work Related to the Thesis

Published Journals (Impact)

- M. Ron, P. Burget, and V. Hlaváč, "Parameter continuity in time-varying Gauss-Markov models for learning from small training data sets", *Information Sciences*, vol. 595, pp. 197–216, 2022, ISSN: 0020-0255. DOI: `https://doi.org/10.1016/j.ins.2022.02.037`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S0020025522001724`
  - Share of the author 85 %.
  - Journal statistics JCR 2020 - IF 6.79: Q1 (11.1 %); AIS 1.33: Q1 (16.7 %)

Submitted Journals (Impact)

- M. Ron, P. Cezner, and P. Burget, "MoDiF: Modular framework for variable-length motif discovery", 2022, Submitted
  - Share of the author 60 %.

Conferences

- M. Ron, P. Burget, and O. Fiala, "Identification of operations at robotic welding lines", in *2015 IEEE International Conference on Automation Science and Engineering (CASE)*, Aug. 2015, pp. 470–476. DOI: `10.1109/COASE.2015.7294124`
  - Share of the author 70 %.

  The paper has been cited in:

  - S. Riazi, K. Bengtsson, R. Bischoff, *et al.*, "Energy and peak-power optimization of existing time-optimal robot trajectories", vol. 2016-November, IEEE, Aug. 2016, pp. 321–327, ISBN: 978-1-5090-2409-4. DOI: `10.1109/COASE.2016.7743423`. [Online]. Available: `http://ieeexplore.ieee.org/document/7743423/`
  - S. Riazi, O. Wigström, K. Bengtsson, *et al.*, "Energy and peak power optimization of time-bounded robot trajectories", *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 2, pp. 646–657, Apr. 2017, ISSN: 1545-5955. DOI: `10.1109/TASE.2016.2641743`
  - P. Yao and K. Zhou, "Application of short time energy analysis in monitoring the stability of arc sound signal", *Measurement*, vol. 105, pp. 98–105, 2017, ISSN: 0263-2241. DOI: `https://doi.org/10.1016/j.measurement.2017.04.015`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S0263224117302294`

- M. Ron and P. Burget, "Stochastic modelling and identification of industrial robots", in *2016 IEEE International Conference on Automation Science and Engineering (CASE)*, Aug. 2016, pp. 342–347. DOI: `10.1109/COASE.2016.7743426`
  - Share of the author 90 %.
- M. Ron and P. Burget, "Density based clustering for detection of robotic operations", in *2017 13th IEEE Conference on Automation Science and Engineering (CASE)*, Aug. 2017, pp. 314–319. DOI: `10.1109/COASE.2017.8256122`
  - Share of the author 90 %.

Book Section

- P. Burget, L. Bukata, P. Šůcha, *et al.*, "Optimisation of power consumption for robotic lines in automotive industry", in *Math for the Digital Factory*, L. Ghezzi, D. Hömberg, and C. Landry, Eds. Cham: Springer International Publishing, 2017, pp. 135–161, ISBN: 978-3-319-63957-4. DOI: `10.1007/978-3-319-63957-4_7`. [Online]. Available: `https://doi.org/10.1007/978-3-319-63957-4_7`
  - The book has 32 citations according to Springer.
  - Authors have equal shares on the section (20 %).

Summer School Proceedings

- L. Jozwiak, R. Stojanovic, and N. Voros, "Edge computing: The BRAINE solution", in *Proceedings of the 3rd Summer School on Cyber-Physical Systems and Internet-of-Things*, 2022, pp. 410–470. DOI: `https://doi.org/10.5281/zenodo.6722220`
  - Authors have equal shares on the section (33 %).

List of Other Candidate's Work

Industrial Research Projects Results

- M. Ron, R. Škoviera, D. Tošner, *et al.*, *Collaborative robotic cell*, Prototype, Accessed: 2022-06-04, Prague, 2021. [Online]. Available: `https://www.isvavai.cz/riv?s=jednoduche-vyhledavani&ss=detail&n=0&h=RIV%2F02673975%3A_____%2F21%3AN0000001`

- M. Ron, H. K. Chvalová, D. Tošner, *et al.*, *Nonsingular trajectory planning*, Software, Accessed: 2022-06-04, Prague, 2021. [Online]. Available: `https://www.isvavai.cz/riv?s=jednoduche-vyhledavani&ss=detail&n=0&h=RIV%2F02673975%3A_____%2F21%3AN0000002`

- M. Ron, D. Tošner, O. Maslikiewicz, *et al.*, *Trifocal scanning head*, Prototype, Accessed: 2022-06-04, Prague, 2020. [Online]. Available: `https://www.isvavai.cz/riv?s=jednoduche-vyhledavani&ss=detail&n=0&h=RIV%2F02673975%3A_____%2F20%3AN0000001`

- M. Ron, P. Cezner, P. Burget, *et al.*, *Behavior models and diagnostic models*, Software, Accessed: 2022-06-04, Prague, 2019. [Online]. Available: `https://www.isvavai.cz/riv?s=jednoduche-vyhledavani&ss=detail&n=0&h=RIV%2F02673975%3A_____%2F19%3AN0000002`

- P. Burget, M. Ron, L. Bukata, *et al.*, *Optimization of robot consumption*, Verified Technology, Accessed: 2022-06-04, Prague, 2014. [Online]. Available: `https://www.isvavai.cz/riv?s=jednoduche-vyhledavani&ss=detail&n=0&h=RIV%2F68407700%3A21230%2F14%3A00227683`

Bibliography

[1] S. Riazi, K. Bengtsson, O. Wigström, E. Vidarsson, and B. Lennartson, “Energy optimization of multi-robot systems”, in *2015 IEEE International Conference on Automation Science and Engineering (CASE)*, Aug. 2015, pp. 1345–1350. DOI: `10.1109/CoASE.2015.7294285`.

[2] L. Bukata and P. Šůcha, “High-level optimisation of robotic lines with respect to power consumption and given production cycle time”, in *27th Conference of the European Chapter on Combinatorial Optimization*, TU Munchen, May 2014.

[3] M. Ron and P. Burget, “Stochastic modelling and identification of industrial robots”, in *2016 IEEE International Conference on Automation Science and Engineering (CASE)*, Aug. 2016, pp. 342–347. DOI: `10.1109/COASE.2016.7743426`.

[4] L. Rabiner and B. Juang, “An introduction to hidden Markov models”, *IEEE ASSP Magazine*, vol. 3, no. 1, pp. 4–16, Jan. 1986, ISSN: 0740-7467. DOI: `10.1109/MASSP.1986.1165342`.

[5] P. L. Ainsleigh, N. Kehtarnavaz, and R. L. Streit, “Hidden Gauss-Markov models for signal classification”, *IEEE Transactions on Signal Processing*, vol. 50, no. 6, pp. 1355–1367, Jun. 2002, ISSN: 1053-587X. DOI: `10.1109/TSP.2002.1003060`.

[6] W. Khalil and E. Dombre, *Modeling, Identification and Control of Robots*, 1st ed. Elsevier Ltd., 2002, ISBN: 978-1-903996-66-9.

[7] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*. Springer, 2009, ISBN: 978-1-84628-641-4.

[8] D. Meike, “Increasing energy efficiency of robotized production systems in automobile manufacturing”, Ph.D. dissertation, Riga Technical University, 1 Kalku Street, Riga, LV-1658, Dec. 2013.

[9] A. Othman, K. Belda, and P. Burget, “Physical modelling of energy consumption of industrial articulated robots”, in *2015 15th International Conference on Control, Automation and Systems (ICCAS)*, Oct. 2015, pp. 784–789. DOI: `10.1109/ICCAS.2015.7364727`.

[10] M. Ron, P. Burget, and O. Fiala, “Identification of operations at robotic welding lines”, in *2015 IEEE International Conference on Automation Science and Engineering (CASE)*, Aug. 2015, pp. 470–476. DOI: `10.1109/COASE.2015.7294124`.

[11] P. Burget, L. Bukata, P. Šůcha, M. Ron, and Z. Hanzálek, “Optimisation of power consumption for robotic lines in automotive industry”, in *Math for the Digital Factory*, L. Ghezzi, D. Hömberg, and C. Landry, Eds. Cham: Springer International Publishing, 2017, pp. 135–161, ISBN: 978-3-319-63957-4. DOI: `10.1007/978-3-319-63957-4_7`. [Online]. Available: `https://doi.org/10.1007/978-3-319-63957-4_7`.

[12] A. M. Gavrovska, M. P. Paskaš, D. Dujković, and I. S. Reljin, “Region-based phonocardiogram event segmentation in spectrogram image”, *10th Symposium on Neural Network Applications in Electrical Engineering, NEUREL-2010 - Proceedings*, pp. 69–72, 2010. DOI: `10.1109/NEUREL.2010.5644108`.

[13] M. Baranski and J. Voss, "Detecting patterns of appliances from total load data using a dynamic programming approach", *Proceedings - Fourth IEEE International Conference on Data Mining, ICDM 2004*, pp. 327–330, 2004. DOI: `10.1109/ICDM.2004.10003`.

[14] M. Weiss, A. Helfenstein, F. Mattern, and T. Staake, "Leveraging smart meter data to recognize home appliances", *2012 IEEE International Conference on Pervasive Computing and Communications, PerCom 2012*, pp. 190–197, 2012. DOI: `10.1109/PerCom.2012.6199866`.

[15] G. Ruzzelli, C. Nicolas, a. Schoofs, and G. M. P. O'Hare, "Real-time recognition and profiling of appliances through a single electricity sensor", *SECON 2010 - 2010 7th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, 2010. DOI: `10.1109/SECON.2010.5508244`.

[16] N Nikolaev, Y Rangelov, V Valchev, and a Marinov, "Technique for indirect analysis of domestic power consumers based on power pattern recognition for smart energy metering", *2013 36th International Convention on Information & Communication Technology Electronics & Microelectronics (MIPRO),*, pp. 971–974, 2013.

[17] A. C. S. D. Queiroz and J. A. F. Costa, "Behavior pattern recognition in electric power consumption series using data mining tools", in *Intelligent Data Engineering and Automated Learning - IDEAL 2012*, H. Yin, J. A. F. Costa, and B. Guilherme, Eds., Natal, Brazil: Springer Verlag, 2012, pp. 522–531.

[18] C. V. Le, C. K. Pang, O. P. Gan, X. M. Chee, D. H. Zhang, M. Luo, H. L. Chan, and F. L. Lewis, "Classification of energy consumption patterns for energy audit and machine scheduling in industrial manufacturing systems", *Transactions of the Institute of Measurement and Control*, vol. 35, no. 5, pp. 583–592, 2012, ISSN: 0142-3312. DOI: `10.1177/0142331212460883`.

[19] L. Simon and K. Hungerbuehler, "Real time Takagi-Sugeno fuzzy model based pattern recognition in the batch chemical industry", in *Fuzzy Systems, 2008. FUZZ-IEEE 2008. (IEEE World Congress on Computational Intelligence). IEEE International Conference on*, Jun. 2008, pp. 779–782. DOI: `10.1109/FUZZY.2008.4630459`.

[20] S. Sunny, P. David, and K. Jacob, "Feature extraction methods based on linear predictive coding and wavelet packet decomposition for recognizing spoken words in Malayalam", in *Advances in Computing and Communications (ICACC), 2012 International Conference on*, Aug. 2012, pp. 27–30. DOI: `10.1109/ICACC.2012.7`.

[21] T. Sledevic and D. Navakauskas, "FPGA based fast Lithuanian isolated word recognition system", in *EUROCON, 2013 IEEE*, Jul. 2013, pp. 1630–1636. DOI: `10.1109/EUROCON.2013.6625195`.

[22] M. Ron, P. Cezner, and P. Burget, "MoDiF: Modular framework for variable-length motif discovery", 2022, Submitted.

[23] A. Mueen, E. Keogh, Q. Zhu, S. Cash, and B. Westover, "Exact discovery of time series motifs", in *Proceedings of the 2009 SIAM International Conference on Data Mining*, Philadelphia, PA: Society for Industrial and Applied Mathematics, Apr. 2009, pp. 473–484, ISBN: 978-0-89871-682-5. DOI: `10.1137/1.9781611972795.41`. [Online]. Available: `https://epubs.siam.org/doi/10.1137/1.9781611972795.41`.

[24] M. K. Das and H.-K. Dai, "A survey of DNA motif finding algorithms", *BMC Bioinformatics*, vol. 8, S21, S7 Dec. 2007, ISSN: 1471-2105. DOI: `10.1186/1471-2105-8-S7-S21`. [Online]. Available: `https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-8-S7-S21`.

[25] R. Wankhedkar and S. K. Jain, “A brief survey on techniques used in discovering time series motifs”, *SSRN Electronic Journal*, 2020, ISSN: 1556-5068. DOI: 10.2139/ssrn.3575372. [Online]. Available: https://www.ssrn.com/abstract=3575372.

[26] D. Pyle, *Data preparation for data mining*, 1999. [Online]. Available: https://www.amazon.com/Preparation-Mining-Kaufmann-Management-Systems/dp/1558605290.

[27] J. Lin, E. Keogh, S. Lonardi, and B. Chiu, “A symbolic representation of time series, with implications for streaming algorithms”, in *Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, ser. DMKD ’03, New York, NY, USA: ACM, 2003, pp. 2–11. DOI: 10.1145/882082.882086. [Online]. Available: http://doi.acm.org/10.1145/882082.882086 (visited on 07/23/2019).

[28] M. Sewwandi, Y. Li, and J. Zhang, “Automated granule discovery in continuous data for feature selection”, *Information Sciences*, vol. 578, pp. 323–343, Nov. 2021, ISSN: 00200255. DOI: 10.1016/j.ins.2021.07.042. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0020025521007325.

[29] N. Tavabi and K. Lerman, *Pattern discovery in time series with byte pair encoding*, 2021. DOI: 10.48550/ARXIV.2106.00614. [Online]. Available: https://arxiv.org/abs/2106.00614.

[30] K. Yang and C. Shahabi, “A PCA-based similarity measure for multivariate time series”, in *Proceedings of the 2Nd ACM International Workshop on Multimedia Databases*, ser. MMDB ’04, New York, NY, USA: ACM, 2004, pp. 65–74, ISBN: 978-1-58113-975-4. DOI: 10.1145/1032604.1032616. [Online]. Available: http://doi.acm.org/10.1145/1032604.1032616 (visited on 07/23/2019).

[31] J. B. d. Souza, V. A. Reisen, G. C. Franco, M. Ispány, P. Bondon, and J. M. Santos, “Generalized additive models with principal component analysis: An application to time series of respiratory disease and air pollution data”, en, *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, vol. 67, no. 2, pp. 453–480, 2018, ISSN: 1467-9876. DOI: 10.1111/rssc.12239. [Online]. Available: https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/rssc.12239 (visited on 07/23/2019).

[32] U. N. Chowdhury, S. K. Chakravarty, and M. T. Hossain, “Short-term financial time series forecasting integrating principal component analysis and independent component analysis with support vector regression”, *Journal of Computer and Communications*, vol. 06, pp. 51–67, 03 2018, ISSN: 2327-5219. DOI: 10.4236/jcc.2018.63004. [Online]. Available: http://www.scirp.org/journal/doi.aspx?DOI=10.4236/jcc.2018.63004.

[33] M. Ali, M. W. Jones, X. Xie, and M. Williams, “TimeCluster: Dimension reduction applied to temporal data for visual analytics”, en, *Vis Comput*, vol. 35, no. 6-8, pp. 1013–1026, Jun. 2019, ISSN: 0178-2789, 1432-2315. DOI: 10.1007/s00371-019-01673-y. [Online]. Available: http://link.springer.com/10.1007/s00371-019-01673-y (visited on 09/19/2019).

[34] H. Qiu, H. T. Lam, F. Fusco, and M. Sinn, *Learning correlation space for time series*, 2018. DOI: 10.48550/ARXIV.1802.03628. [Online]. Available: https://arxiv.org/abs/1802.03628.

[35] M. Linardi, Y. Zhu, T. Palpanas, and E. Keogh, “Matrix profile goes MAD: Variable-length motif and discord discovery in data series”, *Data Mining and Knowledge Discovery*, vol. 34, pp. 1022–1071, 4 Jul. 2020, ISSN: 1384-5810. DOI: 10.1007/s10618-020-00685-w. [Online]. Available: https://link.springer.com/10.1007/s10618-020-00685-w.

[36] H. Sakoe and S. Chiba, “Dynamic programming algorithm optimization for spoken word recognition”, *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 26, no. 1, pp. 43–49, Feb. 1978, ISSN: 0096-3518. DOI: 10.1109/TASSP.1978.1163055.

[37] R. Agrawal, C. Faloutsos, and A. Swami, "Efficient similarity search in sequence databases", en, in *Foundations of Data Organization and Algorithms*, D. B. Lomet, Ed., ser. Lecture Notes in Computer Science, Springer Berlin Heidelberg, 1993, pp. 69–84, ISBN: 978-3-540-48047-1.

[38] Kin-Pong Chan and Ada Wai-Chee Fu, "Efficient time series matching by wavelets", in *Proceedings 15th International Conference on Data Engineering (Cat. No.99CB36337)*, Mar. 1999, pp. 126–133. DOI: 10.1109/ICDE.1999.754915.

[39] Y.-l. Wu, D. Agrawal, and A. El Abbadi, "A comparison of DFT and DWT based similarity search in time-series databases", *Proceedings of the 9th International Conference on Information and Knowledge Management*, Nov. 2000. DOI: 10.1145/354756.354857.

[40] Z. Jiang and K. Liu, "Real time interpretation and optimization of time series data stream in big data", in *2018 IEEE 3rd International Conference on Cloud Computing and Big Data Analysis (ICCCBDA)*, Apr. 2018, pp. 243–247. DOI: 10.1109/ICCCBDA.2018.8386520.

[41] J. D. Drover and N. D. Schiff, "A method for decomposing multivariate time series into a causal hierarchy within specific frequency bands", en, *J Comput Neurosci*, vol. 45, no. 2, pp. 59–82, Oct. 2018, ISSN: 1573-6873. DOI: 10.1007/s10827-018-0691-y. [Online]. Available: https://doi.org/10.1007/s10827-018-0691-y (visited on 07/23/2019).

[42] K. Kalpakis, D. Gada, and V. Puttagunta, "Distance measures for effective clustering of ARIMA time-series", in *Proceedings 2001 IEEE International Conference on Data Mining*, Nov. 2001, pp. 273–280. DOI: 10.1109/ICDM.2001.989529.

[43] A. Savvides, V. J. Promponas, and K. Fokianos, "Clustering of biological time series by cepstral coefficients based distances", *Pattern Recognition*, vol. 41, no. 7, pp. 2398–2412, Jul. 2008, ISSN: 0031-3203. DOI: 10.1016/j.patcog.2008.01.002. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0031320308000137 (visited on 07/23/2019).

[44] D. S. Stoffer, "Detecting common signals in multiple time series using the spectral envelope", *Journal of the American Statistical Association*, vol. 94, no. 448, pp. 1341–1356, Dec. 1999, ISSN: 0162-1459. DOI: 10.1080/01621459.1999.10473886. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/01621459.1999.10473886 (visited on 07/23/2019).

[45] Z. Zimmerman, K. Kamgar, N. S. Senobari, B. Crites, G. Funning, P. Brisk, and E. Keogh, "Scaling time series motif discovery with GPUs: Breaking the quintillion pairwise comparisons a day barrier", in *Proceedings of the ACM Symposium on Cloud Computing*, ACM, Nov. 2019, pp. 74–86, ISBN: 9781450369732. DOI: 10.1145/3357223.3362721. [Online]. Available: https://dl.acm.org/doi/10.1145/3357223.3362721.

[46] N. Castro and P. Azevedo, "Multiresolution motif discovery in time series", en, in *Proceedings of the 2010 SIAM International Conference on Data Mining*, Society for Industrial and Applied Mathematics, Apr. 2010, pp. 665–676, ISBN: 978-0-89871-703-7 978-1-61197-280-1. DOI: 10.1137/1.9781611972801.73. [Online]. Available: https://epubs.siam.org/doi/10.1137/1.9781611972801.73 (visited on 07/23/2019).

[47] C. M. Yeh, Y. Zhu, L. Ulanova, N. Begum, Y. Ding, H. A. Dau, D. F. Silva, A. Mueen, and E. Keogh, "Matrix profile I: All pairs similarity joins for time series: A unifying view that includes motifs, discords and shapelets", in *2016 IEEE 16th International Conference on Data Mining (ICDM)*, Nov. 2016, pp. 1317–1322. DOI: 10.1109/ICDM.2016.0179.

[48] A. Mueen, "Enumeration of time series motifs of all lengths", in *2013 IEEE 13th International Conference on Data Mining*, IEEE, Dec. 2013, pp. 547–556, ISBN: 978-0-7695-5108-1. DOI: 10.1109/ICDM.2013.27. [Online]. Available: http://ieeexplore.ieee.org/document/6729539/.

[49] B. Chiu, E. Keogh, and S. Lonardi, “Probabilistic discovery of time series motifs”, *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Mar. 2003. DOI: 10.1145/956750.956808.

[50] J. Buhler and M. Tompa, “Finding motifs using random projections”, eng, *J. Comput. Biol.*, vol. 9, no. 2, pp. 225–242, 2002, ISSN: 1066-5277. DOI: 10.1089/10665270252935430.

[51] Y. Gao and J. Lin, “Efficient discovery of time series motifs with large length range in million scale time series”, in *2017 IEEE International Conference on Data Mining (ICDM)*, vol. 2017, IEEE, Nov. 2017, pp. 1213–1222, ISBN: 978-1-5386-3835-4. DOI: 10.1109/ICDM.2017.8356939. [Online]. Available: https://ieeexplore.ieee.org/document/8356939/.

[52] ——, “HIME: Discovering variable-length motifs in large-scale time series”, *Knowledge and Information Systems*, vol. 61, pp. 513–542, 1 Oct. 2019, ISSN: 0219-1377. DOI: 10.1007/s10115-018-1279-6. [Online]. Available: http://link.springer.com/10.1007/s10115-018-1279-6.

[53] X. Xi, E. Keogh, C. Shelton, L. Wei, and C. A. Ratanamahatana, “Fast time series classification using numerosity reduction”, in *Proceedings of the 23rd international conference on Machine learning - ICML ’06*, ACM Press, 2006, pp. 1033–1040, ISBN: 1595933832. DOI: 10.1145/1143844.1143974. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1143844.1143974.

[54] Y. Gao and J. Lin, “Discovering subdimensional motifs of different lengths in large-scale multivariate time series”, *2019 IEEE International Conference on Data Mining (ICDM)*, 2019. DOI: 10.1109/icdm.2019.00032.

[55] J. Serra and J. L. Arcos, “Particle swarm optimization for time series motif discovery”, *Knowledge-Based Systems*, vol. 92, pp. 127–137, 2016.

[56] X. Li and J. Lin, “Linear time motif discovery in time series”, in *Proceedings of the 2019 SIAM International Conference on Data Mining*, Society for Industrial and Applied Mathematics, May 2019, pp. 136–144. DOI: 10.1137/1.9781611975673.16. [Online]. Available: https://epubs.siam.org/doi/10.1137/1.9781611975673.16.

[57] S. Torkamani and V. Lohweg, “Survey on time series motif discovery”, *WIREs Data Mining and Knowledge Discovery*, vol. 7, no. 2, e1199, 2017. DOI: https://doi.org/10.1002/widm.1199. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/widm.1199. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/widm.1199.

[58] J. Shieh and E. Keogh, “iSAX: Indexing and mining terabyte sized time series”, in *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD 08*, ACM Press, 2008, p. 623, ISBN: 9781605581934. DOI: 10.1145/1401890.1401966. [Online]. Available: http://dl.acm.org/citation.cfm?doid=1401890.1401966.

[59] J. Lin, E. Keogh, L. Wei, and S. Lonardi, “Experiencing SAX: A novel symbolic representation of time series”, *Data Mining and Knowledge Discovery*, vol. 15, pp. 107–144, 2 Aug. 2007, ISSN: 1384-5810. DOI: 10.1007/s10618-007-0064-z. [Online]. Available: http://link.springer.com/10.1007/s10618-007-0064-z.

[60] A. Kessy, A. Lewin, and K. Strimmer, “Optimal whitening and decorrelation”, *The American Statistician*, vol. 72, no. 4, pp. 309–314, 2018. DOI: 10.1080/00031305.2016.1277159. eprint: https://doi.org/10.1080/00031305.2016.1277159. [Online]. Available: https://doi.org/10.1080/00031305.2016.1277159.

[61] Y. Dodge, D. Cox, D. Commenges, A. Davison, P. Solomon, and S. Wilson, Eds., *The Oxford Dictionary of Statistical Terms*, English, 6th Edition. Oxford: Oxford University Press, Sep. 2006, ISBN: 978-0-19-920613-1.

[62] S. Soldi, V. Beckmann, W. H. Baumgartner, G. Ponti, C. R. Shrader, P. Lubiński, H. A. Krimm, F. Mattana, and J. Tueller, *Long-term variability of AGN at hard X-rays*, Mar. 2014. [Online]. Available: `https://www.aanda.org/articles/aa/abs/2014/03/aa22653-13/aa22653-13.html`.

[63] [dataset] Petr Cezner, *Data sets: Motif implanted into Wiener process*, TestBed website, 2022. [Online]. Available: `https://www.ciirc.cvut.cz/wp-content/uploads/2022/06/Dataset_implanted_motifs.zip`.

[64] A. P. Bradley, "The use of the area under the ROC curve in the evaluation of machine learning algorithms", *Pattern Recognition*, vol. 30, pp. 1145–1159, 7 Jul. 1997, ISSN: 00313203. DOI: `10.1016/S0031-3203(96)00142-2`. [Online]. Available: `https://linkinghub.elsevier.com/retrieve/pii/S0031320396001422`.

[65] R. Briandet, E. K. Kemsley, and R. H. Wilson, "Discrimination of arabica and robusta in instant coffee by Fourier transform infrared spectroscopy and chemometrics", *Journal of Agricultural and Food Chemistry*, vol. 44, no. 1, 170–174, 1996. DOI: `10.1021/jf950305a`.

[66] J. Healey and R. Picard, "Detecting stress during real-world driving tasks using physiological sensors", *IEEE Transactions on Intelligent Transportation Systems*, vol. 6, no. 2, pp. 156–166, 2005. DOI: `10.1109/TITS.2005.848368`.

[67] G. Hebrail and A. Berard, *Individual household electric power consumption data set*, Aug. 2012. [Online]. Available: `https://archive.ics.uci.edu/ml/datasets/Individual+household+electric+power+consumption#`.

[68] M. [dataset] Ron, P. Cezner, and P. Burget, *Data sets: Industrial use case dataset*, TestBed website, 2022. [Online]. Available: `https://www.ciirc.cvut.cz/wp-content/uploads/2022/06/Dataset_industrial_motifs.zip`.

[69] M. Ron and P. Cezner, *MoDiF: Modular framework for variable-length motif discovery*, GitHub, 2022. [Online]. Available: `https://github.com/ronmarti/continuity-preference-hgmm`.

[70] S. Riazi, O. Wigström, K. Bengtsson, and B. Lennartson, "Energy and peak power optimization of time-bounded robot trajectories", *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 2, pp. 646–657, Apr. 2017, ISSN: 1545-5955. DOI: `10.1109/TASE.2016.2641743`.

[71] L. Bukata, P. Šůcha, Z. Hanzálek, and P. Burget, "Energy optimization of robotic cells", *IEEE Transactions on Industrial Informatics*, vol. 13, no. 1, pp. 92–102, Feb. 2017, ISSN: 1551-3203. DOI: `10.1109/TII.2016.2626472`.

[72] H.-Y. Chen, Y.-C. Fan, C.-L. Lai, and H. Chen, "Identifying variable-power appliances in non-intrusive load monitoring systems", *2016 10th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*, pp. 452–457, 2016. DOI: `10.1109/IMIS.2016.124`.

[73] H. Y. Lam, G. S. K. Fung, and W. K. Lee, "A novel method to construct taxonomy electrical appliances based on load signatures", *IEEE Transactions on Consumer Electronics*, vol. 53, no. 2, pp. 653–660, 2007, ISSN: 00983063. DOI: `10.1109/TCE.2007.381742`.

[74] S. K. K. Ng, J. Liang, and J. W. M. Cheng, "Automatic appliance load signature identification by statistical clustering", *Apscom*, pp. 1–6, 2009. DOI: `10.1049/cp.2009.1749`.

[75] S. Mostafavi, B. Futrell, J. Troxler, and R. W. Cox, "Leveraging cloud computing to convert the non-intrusive load monitor into a powerful framework for grid-responsive buildings", in *2016 IEEE International Conference on Big Data (Big Data)*, IEEE, Dec. 2016, pp. 3107–3114, ISBN: 978-1-4673-9005-7. DOI: `10.1109/BigData.2016.7840965`.

[76] K. Michalak, A. Lancucki, and P. Lipinski, "Multiobjective optimization of frequent pattern models in ultra-high frequency time series: Stability versus universality", in *2016 IEEE Congress on Evolutionary Computation (CEC)*, Jul. 2016, pp. 3491–3498. DOI: `10.1109/CEC.2016.7744232`.

[77] J. Shao, K. Hahn, Q. Yang, C. Bohm, A. Wohlschlager, N. Myers, and C. Plant, "Combining time series similarity with density-based clustering to identify fiber bundles in the human brain", *Proceedings - IEEE International Conference on Data Mining, ICDM*, pp. 747–754, 2010, ISSN: 15504786. DOI: `10.1109/ICDMW.2010.15`.

[78] M. Ester, H. P. Krigel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise", in *Proc. of the 2nd International Conference on Knowledge Discorvery and Data Mining*, 1996, pp. 226–231.

[79] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "OPTICS: Ordering points to identify the clustering structure", in *ACM SIGMOD'99 Int. Conf. on Management of Data*, 1999.

[80] C.-S. Hsieh, "Extension of unbiased minimum-variance input and state estimation for systems with unknown inputs", *Automatica*, vol. 45, no. 9, pp. 2149 –2153, 2009, ISSN: 0005-1098. DOI: `http://dx.doi.org/10.1016/j.automatica.2009.05.004`.

[81] S. Gillijns and B. D. Moor, "Unbiased minimum-variance input and state estimation for linear discrete-time systems with direct feedthrough", *Automatica*, vol. 43, no. 5, pp. 934 –937, 2007, ISSN: 0005-1098. DOI: `http://dx.doi.org/10.1016/j.automatica.2006.11.016`.

[82] L. Rozo, P. Jiménez, and C. Torras, "Force-based robot learning of pouring skills using parametric hidden Markov models", in *Robot Motion and Control (RoMoCo), 2013 9th Workshop on*, Jul. 2013, pp. 227–232. DOI: `10.1109/RoMoCo.2013.6614613`.

[83] J. Yang, Y. Xu, and C. S. Chen, "Hidden Markov model-based learning controller", in *Intelligent Control, 1994., Proceedings of the 1994 IEEE International Symposium on*, Aug. 1994, pp. 39–44. DOI: `10.1109/ISIC.1994.367844`.

[84] B. Bona and A. Curatella, "Identification of industrial robot parameters for advanced model-based controllers design", in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, Apr. 2005, pp. 1681–1686. DOI: `10.1109/ROBOT.2005.1570355`.

[85] J. Swevers, W. Verdonck, and J. D. Schutter, "Dynamic model identification for industrial robots", *IEEE Control Systems*, vol. 27, no. 5, pp. 58–71, Oct. 2007, ISSN: 1066-033X. DOI: `10.1109/MCS.2007.904659`.

[86] M. Gautier, A. Janot, and P. O. Vandanjon, "A new closed-loop output error method for parameter identification of robot dynamics", *IEEE Transactions on Control Systems Technology*, vol. 21, no. 2, pp. 428–444, Mar. 2013, ISSN: 1063-6536. DOI: `10.1109/TCST.2012.2185697`.

[87] Q. Guo, W. Perruquetti, and M. Gautier, "On-line robot dynamic identification based on power model, modulating functions and causal Jacobi estimator", in *Advanced Intelligent Mechatronics (AIM), 2014 IEEE/ASME International Conference on*, Jun. 2014, pp. 494–499. DOI: `10.1109/AIM.2014.6878126`.

[88] D. Meike, M. Pellicciari, and G. Berselli, "Energy efficient use of multirobot production lines in the automotive industry: Detailed system modeling and optimization", *IEEE Transactions on Automation Science and Engineering*, vol. 11, no. 3, pp. 798–809, Jul. 2014, ISSN: 1545-5955. DOI: 10.1109/TASE.2013.2285813.

[89] F. IPK, *Realistic robot simulation (RRS)*, Mar. 2016. [Online]. Available: http://www.realistic-robot-simulation.org.

[90] W. Turin, "Unidirectional and parallel Baum-Welch algorithms", *IEEE Transactions on Speech and Audio Processing*, vol. 6, pp. 516–523, 6 1998, ISSN: 10636676. DOI: 10.1109/89.725318. [Online]. Available: http://ieeexplore.ieee.org/document/725318/.

[91] M. Ron, P. Burget, and V. Hlaváč, "Parameter continuity in time-varying Gauss-Markov models for learning from small training data sets", *Information Sciences*, vol. 595, pp. 197–216, 2022, ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2022.02.037. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025522001724.

[92] M. I. Schlesinger and V. Hlaváč, "Recognition of Markovian sequences", in *Ten Lectures on Statistical and Structural Pattern Recognition*, Dordrecht: Springer Netherlands, 2002, ch. 8, pp. 307–312, ISBN: 978-94-017-3217-8. DOI: 10.1007/978-94-017-3217-8_8. [Online]. Available: https://doi.org/10.1007/978-94-017-3217-8_8.

[93] F. S. Al-Anzi and DiaAbuZeina, "Statistical Markovian data modeling for natural language processing", *International Journal of Data Mining & Knowledge Management Process*, vol. 7, 1 2017, ISSN: 2231007X. DOI: 10.5121/ijdkp.2017.7103.

[94] S. Pattnaik, A. K. Nayak, and S. Patnaik, "A semi-supervised learning of HMM to build a POS tagger for a low resourced language", *Journal of Information and Communication Convergence Engineering*, vol. 18, 4 2020, ISSN: 22348883. DOI: 10.6109/jicce.2020.18.4.216.

[95] Y. Wang, H. Yan, H. Zhang, H. Shen, and H. K. Lam, "Interval type-2 fuzzy control for HMM-based multiagent systems via dynamic event-triggered scheme", *IEEE Transactions on Fuzzy Systems*, 2021, ISSN: 19410034. DOI: 10.1109/TFUZZ.2021.3101581.

[96] J. Chen, T. Y. Hsu, C. C. Chen, and Y. C. Cheng, "Monitoring combustion systems using HMM probabilistic reasoning in dynamic flame images", *Applied Energy*, vol. 87, 7 2010, ISSN: 03062619. DOI: 10.1016/j.apenergy.2009.11.008.

[97] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video", in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jul. 2017, pp. 6612–6619, ISBN: 978-1-5386-0457-1. DOI: 10.1109/CVPR.2017.700. [Online]. Available: http://ieeexplore.ieee.org/document/8100183/.

[98] M. Ron and P. Burget, "Density based clustering for detection of robotic operations", in *2017 13th IEEE Conference on Automation Science and Engineering (CASE)*, Aug. 2017, pp. 314–319. DOI: 10.1109/COASE.2017.8256122.

[99] L. Bukata, P. Sucha, Z. Hanzalek, and P. Burget, "Energy optimization of robotic cells", *IEEE Transactions on Industrial Informatics*, vol. PP, no. 99, pp. 1–1, 2016, ISSN: 1551-3203. DOI: 10.1109/TII.2016.2626472.

[100] L. Bukata, P. Sucha, and Z. Hanzalek, "Optimizing energy consumption of robotic cells by a branch & bound algorithm", *Computers & Operations Research*, vol. 102, pp. 52–66, 2019, ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2018.09.012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0305054818302533.

[101] W. Kong, Z. Dong, Y. Xu, and D. Hill, “An enhanced bootstrap filtering method for non-intrusive load monitoring”, *IEEE Power and Energy Society General Meeting*, vol. 2016-November, pp. 1–5, 2016, ISSN: 19449933. DOI: 10.1109/PESGM.2016.7741487.

[102] R. Machlev, Y. Levron, and Y. Beck, “Modified cross-entropy method for classification of events in NILM systems”, *IEEE Transactions on Smart Grid*, vol. PP, no. c, p. 1, 2018, ISSN: 19493053. DOI: 10.1109/TSG.2018.2871620.

[103] A. H. Sabry, F. H. Nordin, A. H. Sabry, and M. Z. Abidin Ab-Kadir, “Fault detection and diagnosis of industrial robot based on power consumption modeling”, *IEEE Transactions on Industrial Electronics*, pp. 1–1, 2019, ISSN: 1557-9948. DOI: 10.1109/TIE.2019.2931511.

[104] W. He, Z. Yan, C. Sun, and Y. Chen, “Adaptive neural network control of a flapping wing micro aerial vehicle with disturbance observer”, *IEEE Transactions on Cybernetics*, vol. 47, no. 10, pp. 3452–3465, Oct. 2017, ISSN: 2168-2275. DOI: 10.1109/TCYB.2017.2720801.

[105] W. He and Y. Dong, “Adaptive fuzzy neural network control for a constrained robot using impedance learning”, *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 4, pp. 1174–1186, Apr. 2018, ISSN: 2162-2388. DOI: 10.1109/TNNLS.2017.2665581.

[106] I. Eski, S. Erkaya, S. Savas, and S. Yildirim, “Fault detection on robot manipulators using artificial neural networks”, *Robotics and Computer-Integrated Manufacturing*, vol. 27, no. 1, pp. 115–123, 2011, ISSN: 07365845. DOI: 10.1016/j.rcim.2010.06.017.

[107] Z. Cao, R. Zhang, Y. Yang, J. Lu, and F. Gao, “Discrete-time robust iterative learning Kalman filtering for repetitive processes”, *IEEE Transactions on Automatic Control*, vol. 61, no. 1, pp. 270–275, 2016, ISSN: 00189286. DOI: 10.1109/TAC.2015.2434073.

[108] F. Wang, Z. Wang, J. Liang, and X. Liu, “Resilient filtering for linear time-varying repetitive processes under uniform quantizations and Round-Robin protocols”, *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 9, pp. 2992–3004, 2018, ISSN: 15498328. DOI: 10.1109/TCSI.2018.2824306.

[109] C. D. McKinnon and A. P. Schoellig, “Experience-based model selection to enable long-term, safe control for repetitive tasks under changing conditions”, *IEEE International Conference on Intelligent Robots and Systems*, pp. 2977–2984, 2018, ISSN: 21530866. DOI: 10.1109/IROS.2018.8593882.

[110] ——, “Learn fast, forget slow: Safe predictive learning control for systems with unknown and changing dynamics performing repetitive tasks”, *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 2180–2187, 2019, ISSN: 23773766. DOI: 10.1109/LRA.2019.2901638. arXiv: 1810.06681.

[111] S. Z. Yong, M. Zhu, and E. Frazzoli, “A unified filter for simultaneous input and state estimation of linear discrete-time stochastic systems”, *Automatica*, vol. 63, pp. 321–329, 2016, ISSN: 00051098. DOI: 10.1016/j.automatica.2015.10.040. [Online]. Available: http://dx.doi.org/10.1016/j.automatica.2015.10.040.

[112] C. S. Hsieh, “Unbiased minimum-variance input and state estimation for systems with unknown inputs: A system reformation approach”, *Automatica*, vol. 84, pp. 236–240, 2017, ISSN: 00051098. DOI: 10.1016/j.automatica.2017.06.037. [Online]. Available: http://dx.doi.org/10.1016/j.automatica.2017.06.037.

[113] E. E. Holmes, *Derivation of an EM algorithm for constrained and unconstrained multivariate autoregressive state-space (MARSS) models*, 2013. DOI: 10.48550/ARXIV.1302.3919. [Online]. Available: https://arxiv.org/abs/1302.3919.

[114] G. Schwarz *et al.*, “Estimating the dimension of a model”, *The annals of statistics*, vol. 6, no. 2, pp. 461–464, 1978.

[115] R. J. Mackay, “Estimating the order of a hidden Markov model”, *Canadian Journal of Statistics*, vol. 30, no. 4, pp. 573–589, 2002. DOI: 10.2307/3316097. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.2307/3316097. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.2307/3316097.

[116] M. Costa and L. D. Angelis, “Model selection in hidden Markov models: A simulation study”, Working Paper, unpublished, Bologna, IT, Dec. 2010, [Online]. Available: http://amsacta.unibo.it/2909/.

[117] R. J. Little and D. B. Rubin, *Statistical analysis with missing data*. John Wiley & Sons, 2019, vol. 793, ISBN: 0471802549.

[118] T. Beckers, J. Umlauft, and S. Hirche, “Stable model-based control with Gaussian process regression for robot manipulators”, *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 3877–3884, Jul. 2017, ISSN: 24058963. DOI: 10.1016/j.ifacol.2017.08.359. arXiv: 1811.06655.

[119] L. E. Baum, T. Petrie, G. W. Soules, and N. Weiss, “A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains”, *Annals of Mathematical Statistics*, vol. 41, pp. 164–171, 1970.

[120] G. Golub, S. Nash, and C. V. Loan, “A Hessenberg-Schur method for the problem AX + XB = C”, *IEEE Transactions on Automatic Control*, vol. 24, pp. 909–913, 6 Dec. 1979, ISSN: 0018-9286. DOI: 10.1109/TAC.1979.1102170. [Online]. Available: http://ieeexplore.ieee.org/document/1102170/.

[121] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks”, in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, Y. W. Teh and M. Titterington, Eds., ser. Proceedings of Machine Learning Research, vol. 9, Chia Laguna Resort, Sardinia, Italy: PMLR, May 2010, pp. 249–256. [Online]. Available: https://proceedings.mlr.press/v9/glorot10a.html.

[122] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT Press, 2012, ISBN: 978-0262018029.

[123] A. Vehtari, A. Gelman, and J. Gabry, “Practical Bayesian model evaluation using leave-one-out cross-validation and WAIC”, *Statistics and Computing*, vol. 27, no. 5, pp. 1413–1432, 2017, ISSN: 15731375. DOI: 10.1007/s11222-016-9696-4. arXiv: 1507.04544.

[124] M. Ron, *Datasets: Robot KR5 and KR210 power consumption*, TestBed website, Accessed: 2021-10-19, 2021. [Online]. Available: https://www.ciirc.cvut.cz/wp-content/uploads/2021/10/Dataset_RobotsEnergy.zip.

[125] M. Balandat, B. Karrer, D. Jiang, S. Daulton, B. Letham, A. G. Wilson, and E. Bakshy, “BoTorch: A framework for efficient Monte-Carlo Bayesian optimization”, in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33, Curran Associates, Inc., 2020, pp. 21 524–21 538. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/f5b1b89d98b7286673128a5fb112cb9a-Paper.pdf.

[126] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh, “The great time series classification bake off: A review and experimental evaluation of recent algorithmic advances”, *Data Mining and Knowledge Discovery*, vol. 31, pp. 606–660, 3 2017. DOI: https://doi.org/10.1007/s10618-016-0483-9.

[127] M. Veloso and D. Vail, *Dataset: Sony AIBO robot surface 1*, UEA & UCR Time Series Classification Repository, Accessed: 2021-10-15, 2011. [Online]. Available: http://www.timeseriesclassification.com/description.php?Dataset=SonyAIBORobotSurface1.

[128] M. Ron, *Continuity preference in hidden Gauss-Markov models*, Github, 2021. [Online]. Available: `https://github.com/ronmarti/continuity-preference-hgmm`.

[129] *Big data processing and artificial intelligence at the network edge*, Accessed: 2022-06-04, 2022. DOI: `10.3030/876967`. [Online]. Available: `https://cordis.europa.eu/project/id/876967/reporting/es`.

[130] L. Jozwiak, R. Stojanovic, and N. Voros, "Edge computing: The BRAINE solution", in *Proceedings of the 3rd Summer School on Cyber-Physical Systems and Internet-of-Things*, 2022, pp. 410–470. DOI: `https://doi.org/10.5281/zenodo.6722220`.

[131] M. Taboga, "Linear combinations of normal random variables", in *Lectures on probability theory and mathematical statistics*, third, vol. Online appendix, Kindle Direct Publishing, 2017. [Online]. Available: `https://www.statlect.com/probability-distributions/normal-distribution-linear-combinations`.

[132] J. Humpherys, P. Redd, and J. West, "A fresh look at the Kalman filter", *SIAM review*, vol. 54, no. 4, pp. 801–823, 2012.

[133] F. A. Graybill, *Matrices with Applications in Statistics*, 2nd. Belmont, CA: Wadsworth, Dec. 1982, ISBN: 9780534980382.

[134] R. A. Johnson, D. W. Wichern, *et al.*, *Applied multivariate statistical analysis*. Pearson London, UK: 2014, vol. 6.

[135] S. Riazi, K. Bengtsson, R. Bischoff, A. Aurnhammer, O. Wigstrom, and B. Lennartson, "Energy and peak-power optimization of existing time-optimal robot trajectories", vol. 2016-November, IEEE, Aug. 2016, pp. 321–327, ISBN: 978-1-5090-2409-4. DOI: `10.1109/COASE.2016.7743423`. [Online]. Available: `http://ieeexplore.ieee.org/document/7743423/`.

[136] P. Yao and K. Zhou, "Application of short time energy analysis in monitoring the stability of arc sound signal", *Measurement*, vol. 105, pp. 98–105, 2017, ISSN: 0263-2241. DOI: `https://doi.org/10.1016/j.measurement.2017.04.015`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S0263224117302294`.

[137] M. Ron, R. Škoviera, D. Tošner, O. Fiala, H. K. Chvalová, O. Maslikiewicz, M. Mejdrechová, G. Šejnová, K. Štěpánová, and M. Vavrečka, *Collaborative robotic cell*, Prototype, Accessed: 2022-06-04, Prague, 2021. [Online]. Available: `https://www.isvavai.cz/riv?s=jednoduche-vyhledavani&ss=detail&n=0&h=RIV%2F02673975%3A_____%2F21%3AN0000001`.

[138] M. Ron, H. K. Chvalová, D. Tošner, O. Fiala, and O. Maslikiewicz, *Nonsingular trajectory planning*, Software, Accessed: 2022-06-04, Prague, 2021. [Online]. Available: `https://www.isvavai.cz/riv?s=jednoduche-vyhledavani&ss=detail&n=0&h=RIV%2F02673975%3A_____%2F21%3AN0000002`.

[139] M. Ron, D. Tošner, O. Maslikiewicz, H. K. Chvalová, and O. Fiala, *Trifocal scanning head*, Prototype, Accessed: 2022-06-04, Prague, 2020. [Online]. Available: `https://www.isvavai.cz/riv?s=jednoduche-vyhledavani&ss=detail&n=0&h=RIV%2F02673975%3A_____%2F20%3AN0000001`.

[140] M. Ron, P. Cezner, P. Burget, and O. Fiala, *Behavior models and diagnostic models*, Software, Accessed: 2022-06-04, Prague, 2019. [Online]. Available: `https://www.isvavai.cz/riv?s=jednoduche-vyhledavani&ss=detail&n=0&h=RIV%2F02673975%3A_____%2F19%3AN0000002`.

[141] P. Burget, M. Ron, L. Bukata, O. Maslikiewicz, P. Cezner, P. Šůcha, and Z. Hanzálek, *Optimization of robot consumption*, Verified Technology, Accessed: 2022-06-04, Prague, 2014. [Online]. Available: `https://www.isvavai.cz/riv?s=jednoduche-vyhledavani&ss=detail&n=0&h=RIV%2F68407700%3A21230%2F14%3A00227683`.