Vrije Universiteit Brussel

Faculteit Wetenschappen en Bio-ingenieurswetenschappen
Vakgroep Computerwetenschappen

Modeling Exoskeleton-Assisted Human Motion with Gaussian Processes

Proefschrift ingediend met het oog op het behalen van de graad van
Master of Science in de Ingenieurswetenschappen: Computerwetenschappen

Timothy Verstraeten

Promotor: Prof. Dr. Ann Nowé
Begeleider: Anna Harutyunyan

Augustus 2015

Vrije Universiteit Brussel

Faculty of Science and Bio-Engineering Sciences
Department of Computer Science

Modeling Exoskeleton-Assisted Human Motion with Gaussian Processes

Thesis submitted in partial fulfillment of the requirements for the degree of
Master of Science in de Ingenieurswetenschappen: Computerwetenschappen

Timothy Verstraeten

Promotor: Prof. Dr. Ann Nowé
Advisor: Anna Harutyunyan

August 2015

Abstract

The recent development in robotics and human motion piqued interest in researching dynamic human-interactive devices. Specifically, many prototypical designs for exoskeletons, which provide physical assistance, have been proposed. These devices give support by applying strength in the direction the wearer moves to, in order to meet the motion requirements. However, current prototypes require the calibration of the actuators in order to provide person-specific support. In this thesis, we aim to set up a theoretical exoskeleton-assistance framework in which we learn the required support using Gaussian processes, rather than calibrating it ourselves. We introduce a novel method, called SEAM, to learn support in an on-line environment. Additionally, we explore the capabilities of an "all-in-one" support model, which considers the aggregation of multiple activities in one model. In this study, we found that, although conceptually applicable, the generative aspect of SEAM prevents a stable series of predictions, thus failing to learn a proper support model in an on-line environment. We also found that, even though a single-task support model performs significantly better than a multi-task support model in terms of predictive accuracy, the latter provides a much more compact and less redundant approach. Moreover, using a multi-task support model allows for extrapolation to other unseen tasks. Overall, this study provides useful insights in terms of modeling support provided by exoskeletons.

Declaration of Originality

I hereby declare that this thesis was entirely my own work and that any additional sources of information have been duly cited. I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this thesis has not been submitted for a higher degree to any other University or Institution.

Acknowledgements

First and foremost, I like to express my appreciation and gratitude to my thesis advisor Anna Harutyunyan for the encouragement and the weekly interesting discussions. I would also like to thank my promotor prof. dr. Ann Nowé for giving me the opportunity to perform research in this topic, and more importantly, the field of Artificial Intelligence. Additionally, I wish to thank my fellow scientists and friends, for listening to my ideas and providing useful feedback on them. Finally, a big thank you to my parents, for listening to my ideas and nodding.

Contents

1 Introduction
- 1.1 Exoskeletons and Human Motion . . . 2
- 1.2 Research Question and Contributions . . . 3
- 1.3 Thesis Structure . . . 5

2 Gaussian Processes
- 2.1 Marginal Likelihood . . . 10
- 2.2 Hyperparameters . . . 11
- 2.3 Positive Definite Kernels . . . 12
  - 2.3.1 Cholesky Update Rule and Process Approximations . . . 13
- 2.4 Multiple Outputs . . . 15
- 2.5 Dynamic Process . . . 16

3 Models
- 3.1 Trajectories . . . 23
  - 3.1.1 Normal . . . 23
  - 3.1.2 Weak . . . 24
- 3.2 Support . . . 24
- 3.3 Scale-Dependent Transitions . . . 25
  - 3.3.1 Velocity . . . 27
  - 3.3.2 Data Augmentation . . . 27

4 Learning
- 4.1 Scale Extraction and Adaptation Method . . . 29
  - 4.1.1 Separating Scale from Deviation Information . . . 30
  - 4.1.2 Confident Scale Information . . . 32
  - 4.1.3 Running Average over Scales . . . 33
  - 4.1.4 Alignment . . . 34
  - 4.1.5 Assumptions . . . 34
- 4.2 Support . . . 35
  - 4.2.1 Multi-Task . . . 35

5 Experiments
5.1 Evaluation 38
5.2 Data 39
5.3 SEAM Parameters 39
5.4 SEAM Assessment 41
5.4.1 Results 43
5.4.2 Discussion 48
5.5 Single-Task Support Assessment 49
5.5.1 Results 50
5.5.2 Discussion 53
5.6 Multi-Task Support Assessment 55
5.6.1 Results 55
5.6.2 Discussion 58
5.7 Children's Gait Support 59

6 Conclusion
6.1 Future Work 64

List of Symbols

I	identity matrix
$PDF_{[\mathbf{f}]}$	probability density function of random variables $\mathbf{f}$
U	Cholesky factor
X, X^*	matrices of training and test input locations
$\delta_p^{a^{-1}}(\cdot), \delta_p^a(\cdot)$	support and reduction process of person p for activity a
ϵ	white Gaussian additive noise
$\eta^a(\cdot)$	normal process for activity a
$\frac{\partial}{\partial x}$	partial derivative w.r.t. x
$\log$	natural logarithm
$\mathbb{E}$	expected value
$\mathcal{N}, \mathcal{GP}$	Gaussian distribution, Gaussian process
$\mathcal{O}$	Big-O notation
$\boldsymbol{\theta}, \boldsymbol{\theta}_\mathbf{c}$	vector of hyperparameters, vector of correlation hyperparameters for multi-output kernel
$\boldsymbol{\zeta}_\mathbf{t}$	vector of latent states at time stamps/indices $\mathbf{t}$
$\mathbf{f}, \mathbf{f}^*$	vectors of training and test latent function values
$\mathbf{s}_\mathbf{t}^\gamma$	observed state drawn from process γ at time stamp/index t
$\mathbf{t}$	vector of time stamps
$\mathbf{v}_\mathbf{t}^\gamma$	observed velocity vector at time t, associated with process γ
$\mathbf{y}$	vector of observed function values
$\omega_p^a(\cdot)$	weak process of person p for activity a
σ	standard deviation
$\widetilde{\delta}_p^a(\cdot), \delta_p^a(\cdot)$	deviation process and reduction process of person p for activity a
$\widetilde{x}$	approximation of x
$a \mid b$	random variable a conditioned on b
$conf(\cdot)$	function indicating the confidence of a scale
d, d_{out}	number of input dimensions, number of output dimensions
$f(\cdot)$	random process
$h, \boldsymbol{\lambda}$	output scale, vector of input scales

$k(\mathbf{x}, \mathbf{x'}), K(X, X')$	covariance kernel function applied to inputs $\mathbf{x}, \mathbf{x'}$, covariance matrix built from inputs X, X'
l, l'	label associated with an output dimension
$m(\mathbf{x}), \mathbf{m}(X)$	mean function applied to input location $\mathbf{x}$, mean vector built from inputs X
n	number of observations
$p(\cdot), p(\cdot \mid \cdot)$	marginal probability, conditional probability
r^{γ}, ρ^{γ}	observed and model scale for process γ
$size_W$	window size used in SEAM
$thresh_{conf}$	confidence threshold used in SEAM
v_{proj}, v_{rej}	projected and rejected vector of v
$\lvert M \rvert$	determinant of matrix M
$\lvert\lvert \cdot \rvert\rvert, \lvert\lvert \cdot \rvert\rvert_2$	2-norm of a vector

1

INTRODUCTION

One of the intriguing aspects that separates us humans from other animals is our consistent and meaningful usage of tools. Over many centuries, we came up with new and better ideas to make physical labor easier. One of the first creations that does this, is the 'wheel', which many consider one of greatest inventions in human history. This particularly simply shaped piece of work provides us with many applications to help us do labor that requires human strength. For builders, the focus of using the wheel was set on constructing various tools for transportation of objects, such as wheel barrows and pulleys. These gave a boost in both productivity and life expectancy, as building became less physically restraining.

A more recent application of the wheel is the auto mobile or car, which first appeared in Europe during the Industrial Revolution. After it became available to the consumer market, people could transport 'themselves' by providing negligible energy, as is not the case with a bicycle, for example. Transporting humans efficiently is a necessity for productive large-scale cooperations. As our broad tapestry of civilizations is evolving towards one large community, research and development in this area is still active and a booming business.

However, for everyday activities, such as walking, the development of transportation tools has been left unattended for a long time, making it unevolved in its current state. This is mainly due to the fact that the majority of people can do these tasks without the need for support. Yet, people who are not capable to meet certain motion task requirements, such as walking or standing up, also need the opportunity to perform these activities.

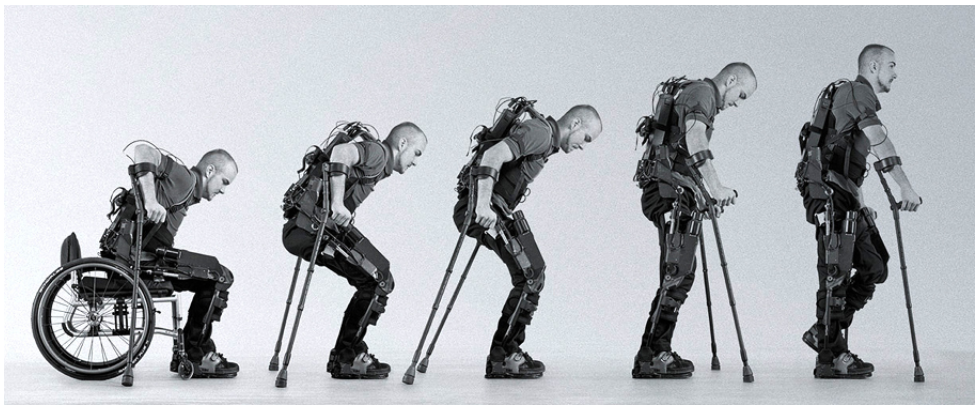


Figure 1.1: Agro's ReWalk exoskeleton that supports paraplegics in everyday activities such as walking and standing up. (Medical Expo, 2014)

1.1 Exoskeletons and Human Motion

Tools that currently are widely available for this purpose, comprises wheel chairs, crutches and prosthetics, which allow people with motor dysfunctions in their legs, or *paraplegics*, to transport themselves using arm strength. Even though these are useful and helpful, we are far from making an analogy to a car, in the sense that cars require little physical effort to make their complex system of machinery drive at fast speed.

A piece of technology invented with this idea in mind, is the powered exoskeleton (Figure 1.1). It can be seen as the motorized counterpart of prosthetics, and aims to provide better and automatic leg support to paraplegics. However, improvements to such devices are still being researched, as it is difficult to create an exoskeleton that is light and flexible to move around with, yet has to support its own weight and that of the wearer.

Therefore, the analysis of human motion is crucial in order to reconstruct the necessary parameters that characterizes it. One solution to this problem is Inverse Kinematics (IK), in which we apply the kinematics equations to derive the joint angles and torques from a pose (Tolani et al., 2000). This approach is the most common, as the equations are physically sound and entirely determined, and allows us to extrapolate to desired poses based on a small movement of the wearer, providing support in the activity.

However, the desired poses are not always those indicated by the wearer. To provide reliable pose transitions, one has to 'model' human motion. This domain is hard, due to its non-linear state space and degrees of freedom of human movements. Common approaches that tackle these problems are Hidden Markov Models (HMM) (Markov, 1906; Rabiner, 1989) and Gaussian Processes

(GP) (Rasmussen and Williams, 2006), which are instances of Bayesian inference methods (Hastie et al., 2009). These methods induce a preference bias over the hypothesis space in the form of Bayesian probabilities and manipulate them according to evidential information. Bayesian methods are known to be flexible, data-efficient and fundamentally sound, which makes them appropriate for non-linear high-dimensional state spaces. Specifically, GPs are widely applied to time-series regression problems (Roberts et al., 2012; Cunningham et al., 2012), and are suitable for modeling spatio-temporal data, making them appropriate for our human motion setting. This has been done by Wang et al. (2008) in the field of computer vision, where pose transitions for a walking human are generated from such a process.

1.2 Research Question and Contributions

In this thesis, we set up a general theoretical framework for exoskeletal assistance, in order to explore the applicability of GPs to model support, i.e., adjusting capability gaps (i.e., "weaknesses" in a motion trajectory) in order to meet a specific motion task requirement. This implies researching and constructing possible support models and methods to apply them accordingly in an on-line environment.

We attempt to generalize assistance to multiple tasks and grasp the correlations between them in order to define a unified concept of the impairment under observation. This could yield a more accurate, yet compact support model, rather than redundantly modeling each motion task separately.

We also propose a novel method to learn a support model in an on-line fashion, which we call the Scale Extraction and Adaptation Method (SEAM). This allows us to align the trajectory under observation with a predicted desired sequence of poses, generated from a process that models movements suitable to adequately perform a specific-motion task.

Additionally, we introduce the concept of multi-task support, which captures the capability gaps over multiple activities. As the same motor dysfunction is the cause for all these gaps, they should be highly correlated, which means we can transfer support information between tasks. Moreover, having a multi-task support model is easier to maintain and extend than managing multiple single-task support models. A general multi-task support model also should be able to extrapolate to unknown tasks performed by the person for which the support model has been constructed. This is useful for an exoskeleton, as it could encounter specific motion tasks for which it has not been trained on yet.

Even though the framework should be applicable to any type of physical disability, we specifically focus on assistance in the context of paraplegia, which is a dysfunction in the lower limb motors. To support the utility and contribu-

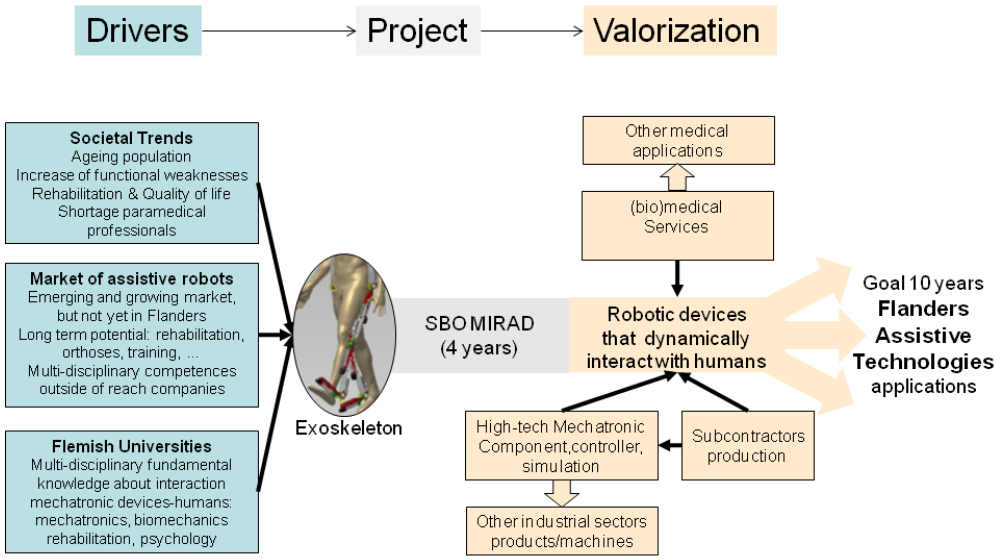


Figure 1.2: Plan and contributions of the MIRAD project. (MIRAD, 2014)

tions of our framework in this context, we specifically refer to the MIRAD project (MIRAD, 2014), as their goals are in line with ours (see Figure 1.2). MIRAD, which stands for *An Integrated **M**ethodology to Bring **I**ntelligent **R**obotic **A**ssistive **D**evices to the User*, concerns itself with the design of a lower-limb exoskeleton (see Figure 1.3). It is initiated due to the lack of paramedics and increase in motor dysfunctions. Additionally, its long-term goal is to establish development of robotic devices which can perform dynamic human interaction, which can be used in other areas, such as the other medical areas and the industrial sector.

Within the MIRAD project, Afschrift et al. (2014) attempted to model 'support' by setting up a *musculoskeletal* model that captures various levels of weakness in the muscles. Even though the method is different, it is similar to what we attempt to do. However, this research is performed from a biomedical point of view, meaning that there exists a direct mapping between aspects of their method and the actuators present in the exoskeleton. In contrast, we attempt to set up a theoretical support framework from a more general perspective, which allows us to gain clear insights into the various aspects required to provide exoskeletal assistance. Additionally, accepting this paradigm allows us to disconnect the support framework from the field of human motion and apply it to other domains.

1.3 Thesis Structure

In Chapter 2, we discuss variants of Gaussian processes and properties useful to our support framework, how we can decrease computational cost and ways to capture correlations amongst multiple processes.

Chapter 3 lists formal definitions and explanations of models, their instantiations and their interrelationships.

In Chapter 4, we explain the learning techniques used to achieve a trained support model. We cover how we can map an observed motion trajectory point-wise with its desired generated counterpart using SEAM, and its theoretical and practical applicability. Additionally, we describe the specifics of a multi-task support model, which captures various capability gaps over multiple physical activities.

In Chapter 5, we set up our experimental environment and present the results for each test case. We assess the performance of our novel method SEAM and our support models independently, after which we evaluate the application of a single-task support model during SEAM. We also present a small proof of concept to indicate the practical applicability of our methods.

In Chapter 6, we discuss the results shown in the previous chapter and the insights gained from them.

We conclude the thesis in Chapter 7, mentioning possibilities for future work.

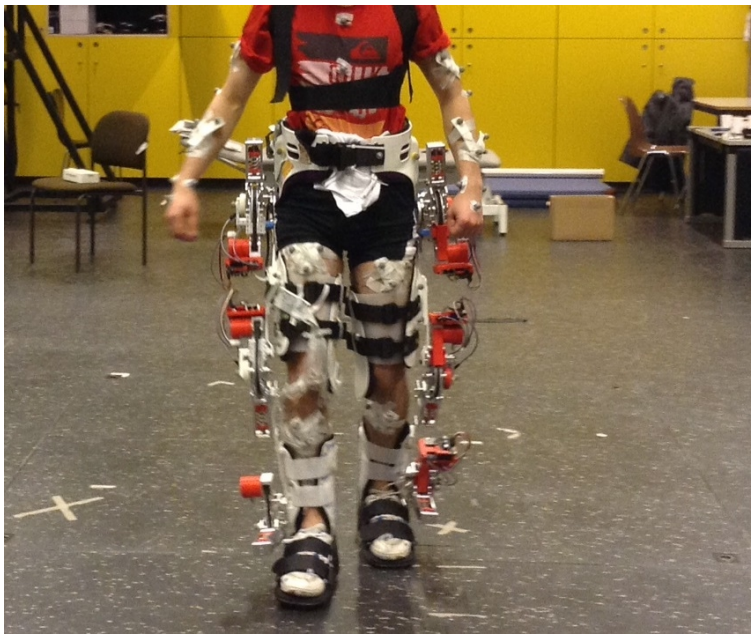


Figure 1.3: First prototype of the MIRAD exoskeleton being tested. (MIRAD, 2014)

2

GAUSSIAN PROCESSES

A Gaussian process (GP) is a generalization of a multivariate normal distribution. Instead of describing a finite vector of random variables, it characterizes a distribution over functions, which can be seen as an uncountably infinite number of random variables. Moreover, each finite set of those variables has a joint normal distribution. Rasmussen and Williams (2006) present a general and vast overview of GPs and their properties.

A GP is defined by a prior mean function $m(\cdot)$ and covariance kernel function $k(\cdot, \cdot)$.

$$\begin{aligned} m(\mathbf{x}) &= \mathbb{E}[f(\mathbf{x})] \\ k(\mathbf{x}, \mathbf{x}') &= \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))] \end{aligned} \tag{2.1}$$

where $\mathbf{x}, \mathbf{x}'$ are input locations and $f(\cdot)$ is a random process. A random process is a mapping from a (infinite) set of input locations to a set of random variables. In other words, a process is similar to a function, but instead of mapping to a single value, it maps to a distribution of values.[1] We denote that $f(\cdot)$ follows a Gaussian process (i.e., each finite set of random variables is jointly normally distributed) with mean $m(\cdot)$ and covariance kernel function $k(\cdot, \cdot)$ as

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \tag{2.2}$$

[1] Note that the word 'process' has two different meanings in this context. While a GP describes a set of functions, $f(\cdot)$ 'is' described by a GP. Even though this distinction is small and mostly irrelevant, we refer to the former and latter case as respectively a GP and random process.

A common choice for a mean function is the constant zero-valued function $m_0(\cdot)$, as it is often hard to include prior knowledge in terms of a mean. However, there are several popular choices for the covariance kernel. One of them is the squared exponential (SE) kernel $k_{SE}(\cdot,\cdot)$,

$$k_{SE}(\mathbf{x},\mathbf{x'}) = h^2 \exp(-\sum_{i=1}^{d} \frac{(\mathbf{x}_i - \mathbf{x'}_i)^2}{\boldsymbol{\lambda}_i^2}) \tag{2.3}$$

where d is the number of dimensions of an input, $\mathbf{x}_i$ the i^{th} dimension of input parameter $\mathbf{x}$, $\boldsymbol{\lambda}_i$ the input/length scale for the i^{th} dimension and h the output scale. The SE kernel should be used when inputs that are close to one another are highly co-varying, and vice versa, as its value increases when its inputs are closer to each other, which is controlled by the length scale $\boldsymbol{\lambda}$. Explanations about the impact of its scale parameters is given in Section 2.2. Note that $k_{SE}(\cdot,\cdot)$ is indeed a valid covariance kernel (Fasshauer, 2011), which means that every covariance matrix constructed from this kernel is invertible (see Section 2.3). From now on, we consider GPs to be characterized by the mean function $m_0(\cdot)$ whenever unspecified.

As mentioned before, each finite set of random variables has a joint normal distribution. The prior distribution is given by

$$\mathbf{f} \mid X \sim \mathcal{N}(\mathbf{m}(X), K(X,X)) \tag{2.4}$$

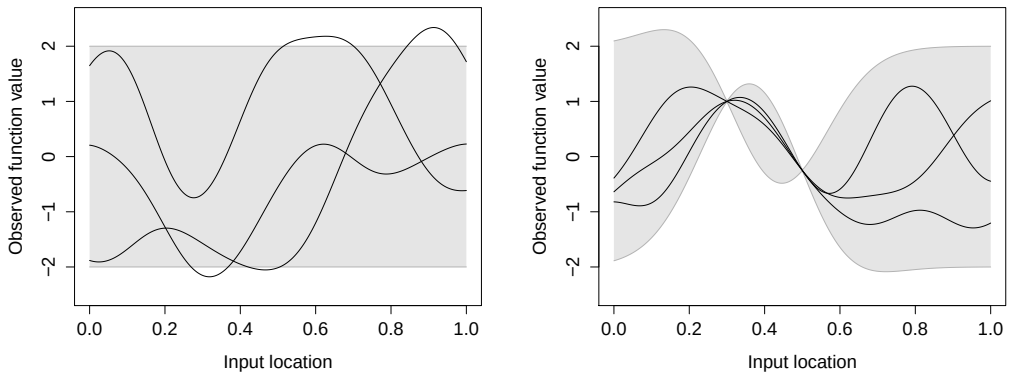
with a probability density function (PDF)

$$PDF_{[\mathbf{f} \mid X]}(\mathbf{z}) = \frac{1}{\sqrt{(2\pi)^d |K(X,X)|}} exp(-\frac{1}{2}(\mathbf{z}-\mathbf{m}(X))^T K(X,X)^{-1}(\mathbf{z}-\mathbf{m}(X))) \tag{2.5}$$

while the posterior distribution is defined as

$$\begin{aligned} &\mathbf{f}^* \mid X^*, X, \mathbf{y} \sim \mathcal{N}(\mu_{\mathbf{f}^*}, \Sigma_{\mathbf{f}^*}), \text{ with} \\ &\qquad \mu_{\mathbf{f}^*} = \mathbf{m}(X^*) + K(X^*,X)K(X,X)^{-1}(\mathbf{y} - \mathbf{m}(X)) \\ &\qquad \Sigma_{\mathbf{f}^*} = K(X^*,X^*) - K(X^*,X)K(X,X)^{-1}K(X,X^*) \end{aligned} \tag{2.6}$$

where X, X^* are matrices of distinct input locations, $\mathbf{f}, \mathbf{f}^*$ their corresponding latent function values and $\mathbf{y}$ observed function values at input location X. The vector $\mathbf{m}(X)$ and matrix $K(X,X^*)$ are respectively the mean vector for $\mathbf{f}$ (i.e., $m(\cdot)$ applied for X) and the covariance matrix between $\mathbf{f}$ and $\mathbf{f}^*$ (i.e., $k(\cdot,\cdot)$ applied for X and X^*). A step-by-step explanation for obtaining the mean and covariance matrix of the posterior distribution is provided by Rasmussen and Williams (2006).



(a) Prior (b) Posterior

Figure 2.1: Three sample functions drawn from a prior (left) and a posterior distribution (right), interpolated at a 100 equidistant variables, of a GP with mean m_0 and covariance kernel k_{SE}. The posterior distribution is conditioned on the observations at 0.3 and 0.5, which are perceived to be 1 and -0.25. The gray areas are the 95% confidence intervals per point.

We can sample from these distributions to retrieve a vector $\mathbf{y}$ of function values at input locations X. Interpolating linearly between these points gives us an approximation of a function that could be drawn from the GP. In Figure 2.1a and 2.1b, three functions drawn from respectively a prior and posterior distribution at 100 equidistant inputs are shown. In the latter, sampled functions are forced to go through the observed points. Moreover, the variances of variables close (in terms of the input domain) to the observed points are small, as they are highly co-varying and can thus be for the most part inferred by these observations. Note that even though in this case, in which we are certain about the observed values (i.e., all sampled functions go through those points), uncertainty on variables can be modeled by incorporating noise in the covariance kernel, such that our process is not fully constrained on the observed function values.

We can apply GPs to various types of problems, such as classification (Williams and Barber, 1998) and regression (Williams and Rasmussen, 1996). As mentioned earlier, we can compute a posterior distribution of a GP, conditioned on observed points and interpolate, or regress, between them. In contrast to other regression models, such as the commonly used linear model, a GP places a preference bias (i.e., the prior) over a function space, rather than a restriction bias (e.g., "only consider linear functions"). Therefore, a GP is more general and flexible in what it can model than a linear regression model (Rasmussen, 2004). Just like many

other regression models, we can train a GP by providing a set of observed data points, called the *training set*, through which the process should regress. Given this finite set, we can model the posterior normal distribution for a set of input locations, called the *test set*, as shown in Equation 2.6.

Even though GPs can model non-linear functions, in contrast to most other regression models, there are some limitations and considerations to keep in mind. First, the main assumption made is that the latent variables are normally distributed. This is in practice not always the case. However, more complex models can be built by combining multiple GPs (Rosenblatt, 2000; Rasmussen, 2000). Secondly, it is not always clear what type of prior should be induced on the function space. This requires domain knowledge and should be designed by experts for optimal usage, as the choice of prior highly influences the model's accuracy.

2.1 Marginal Likelihood

In a linear regression model, the function space is parametrized to capture linear functions. In this case, the parameters are the weight vector $\mathbf{w}$ in the formula $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$.

However, a GP is nonparametric, in the sense that the mapping from the inputs X to their predictive values $\mathbf{y}$ does not depend on parameters of the model. We show this by integrating out these parameters from the likelihood, which denotes how 'likely' our observations are, given the input locations and model parameters imposed on our GP. This procedure is called *marginalization*. The marginalization of the likelihood happens as follows:

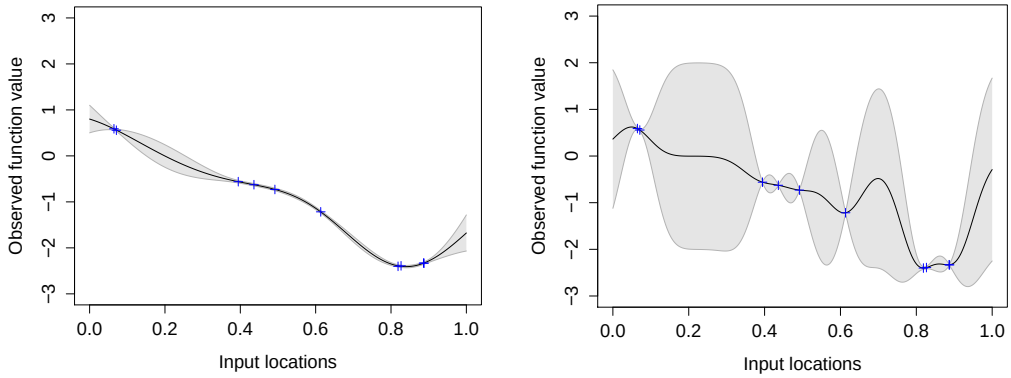
$$p(\mathbf{y} \mid X, \boldsymbol{\theta}) = \int p(\mathbf{y} \mid \mathbf{f}, X, \boldsymbol{\theta}) p(\mathbf{f} \mid X, \boldsymbol{\theta}) d\mathbf{f} \tag{2.7}$$

where X are inputs, $\mathbf{f}$ their latent function values, $\mathbf{y}$ the predictive values and $\boldsymbol{\theta}$ a vector of covariance parameters. The model parameters we attempt to marginalize out are the latent function values $\mathbf{f}$. In order to do this, we require densities for the prior $\mathbf{f} \mid X, \boldsymbol{\theta}$ and likelihood $\mathbf{y} \mid \mathbf{f}, X, \boldsymbol{\theta}$. The former is already formalized in Equation 2.5. We further impose Gaussian white additive noise ϵ (which we consider part of $\boldsymbol{\theta}$) on the latent function values, such that

$$\begin{aligned} y =& f(\mathbf{x}) + \epsilon \text{ with} \\ & \epsilon \sim \mathcal{N}(0, \sigma^2) \end{aligned} \tag{2.8}$$

where $f(\mathbf{x})$ a latent function value, y its corresponding observed value and σ^2 a variance. Using this, we can obtain the Gaussian distribution that describes the likelihood

$$\mathbf{y} \mid \mathbf{f}, X, \boldsymbol{\theta} \sim \mathcal{N}(\mathbf{f}, \sigma^2 I) \tag{2.9}$$



(a) $\lambda = 0.20$ (b) $\lambda = 0.05$

Figure 2.2: Mean and variance plotted for 100 equidistant test samples, conditioned on 10 random training points. The test samples are drawn from a GP with an input scale of λ.

from which we can calculate the PDF. Using these two PDFs, we can solve the integration in Equation 2.7 to retrieve the marginal likelihood $p(\mathbf{y} \mid X, \boldsymbol{\theta})$.

2.2 Hyperparameters

Even though we consider our model to be nonparametric, we still condition our marginal likelihood on parameters $\boldsymbol{\theta}$ in Equation 2.7. These refer to the parameters of the covariance kernel and likelihood function (and mean function, which is not the case for us, considering we use $m_0(\cdot)$), such as the input and output scales of the $k_{SE}(\cdot, \cdot)$ kernel presented in Equation 2.3. However, as they characterize the prior and are not directly placed on the function space, these are independent of the underlying parametric model we marginalized. We refer to these as *hyperparameters*.

These hyperparameters can have a large impact on the model. In the SE kernel $k_{SE}(\cdot, \cdot)$, the input (or length) and output scale characterize respectively the desired magnitude of the input and output values of a function drawn from the GP. Figure 2.2 shows the impact of different input scales λ. We can see that the plotted function on the left has less fluctuations than the one on the right. This is due to the higher length scale, which means that distinct variables are considered less influential towards each other. Thus, the kernel associated with the left figure is less 'short-sighted' than the one considered in the right figure. This is also visible in terms of variance, as the GP becomes more uncertain about unobserved

variables for smaller length scales. Note once more, that these scales are parameters of the covariance kernel and not of the underlying parametric models, thus characterize the stochastic process and not an actual function itself, which means that functions with different length scales are still possible, but less probable.

As different hyperparameters give us distinct behaviours, it is therefore important for solving a regression problem that we can optimize these hyperparameters w.r.t. our domain of application. We do this by choosing the model for which the observations are most 'likely' to occur given the model. This means we must optimize the likelihood of our observations (presented in Equation 2.7) w.r.t. hyperparameters $\boldsymbol{\theta}$. We attempt to minimize the equivalent, simpler and computationally stable negative log-likelihood, expressed as

$$-\log p(\mathbf{y} \mid X, \boldsymbol{\theta}) = \frac{1}{2}(\mathbf{y}^T K(X,X)^{-1}\mathbf{y} + \log|K(X,X)| + n\log 2\pi) \tag{2.10}$$

with n the number of training points.

There are various approaches to solve this optimization problem. One simple method is *conjugate gradient descent*, which is an iterative method that gradually alters the solution towards the gradient of the objective function in order to find a local minimum. In our context, this means we require the derivative of the negative log-likelihood function w.r.t. a hyperparameter $\boldsymbol{\theta}_i$, which is expressed as

$$\begin{aligned} -\frac{\partial}{\partial \boldsymbol{\theta}_i}\log p(\mathbf{y} \mid X, \boldsymbol{\theta}) = &\frac{1}{2}\mathbf{y}^T K(X,X)^{-1}\frac{\partial K(X,X)}{\partial \boldsymbol{\theta}_i}K(X,X)^{-1}\mathbf{y} \\ &+ \frac{1}{2}tr(K(X,X)^{-1}\frac{\partial K(X,X)}{\partial \boldsymbol{\theta}_i}) \end{aligned} \tag{2.11}$$

Adjusting different $\boldsymbol{\theta}_i$ at a time towards the direction of the gradient brings us closer towards a local minimum. Note that this method does not provide us with the 'global' minimum, but rather the local one, which may be suboptimal. Moreover, as the approach optimizes the objective w.r.t. a data set $(X, \mathbf{y})$, one must be cautious of overfitting, which is the phenomenon of modeling noise in the training data instead of the underlying signal. A large, representative data set or limiting the number of iterations could generally prevent this. More information about this method can be found in (Shewchuk, 1994).

2.3 Positive Definite Kernels

So far, we assumed that the matrices $K(X,X)$, given in Equations 2.6, 2.10 and 2.11, are indeed invertible. Even though it is an intrinsic property of a covariance matrix to be invertible, there is no guarantee that the kernels we use would

give us a covariance matrix. In order have a valid covariance kernel, it needs to be *positive semi-definite* (PSD).

Definition 2.1. *A matrix K is PSD if and only if*

$$\forall \mathbf{v} \neq \mathbf{0} \colon \mathbf{v}^T K \mathbf{v} \geq 0$$

A kernel k is PSD if and only if every covariance matrix $K(X, X)$ for inputs X constructed from k is PSD.

Given this definition, we have the following theorem.

Theorem 2.1. *Any PSD matrix is invertible.*

Proof. Every PSD matrix K has a Cholesky decomposition $K = U^T U$ (Schnabel and Eskow, 1990). Moreover, the Cholesky factor U is an upper triangular matrix with non-negative diagonal elements. Therefore, the linear systems of equations

- $UM_1 = I$ (2.12)
- $U^T M_2 = M_1$ (2.13)

can be solved uniquely, using respectively back and forward substitution, leaving us with $M_2 = K^{-1}$.

□

Thus, in order to invert the matrices $K(X, X)$ in our posterior distribution and marginal likelihood, it is sufficient that the kernels from which they are constructed, are PSD.

2.3.1 Cholesky Update Rule and Process Approximations

In Equations 2.12 and 2.13, we use back and forward substitution on the factors of the *Cholesky decomposition* (Schnabel and Eskow, 1990) of matrix K, which can be considered as the square root of a matrix. This is a stable method for matrix inversion and is widely used in case of PSD matrices. However, the complexity of this approach is of the same order as the commonly used Gauss-Jordan elimination for matrix inversion. Even though the back and forward substitutions methods both have a quadratic complexity, the Cholesky decomposition still requires $O(n^3)$.

There are various approximation methods to counter this expensive operation. One of them introduces the concept of inducing inputs (Candela et al., 2007), which are a fixed selective set of inputs Z on which covariances between other

entries should be induced. Candela et al. (2007) show that Subset of Regressors (SoR) approximation algorithm, which is a method based on 'inducing inputs', is equivalent to using a covariance kernel $k_{SoR}(\cdot,\cdot)$ defined as follows:

$$k_{SoR}(\mathbf{x},\mathbf{x}') = \mathbf{k(x,Z)}K(Z,Z)^{-1}\mathbf{k(Z,x')} \tag{2.14}$$

where $\mathbf{k(x,Z)}$ is a vector obtained by applying the covariance kernel $k(\cdot,\cdot)$ to input $\mathbf{x}$ and inducing inputs Z, while $K(Z,Z)$ is the matrix constructed by applying this kernel to the inducing inputs. We can see that covariances amongst variables for which the inputs are not in Z, cannot be expressed directly, which means covariance information has to be transfered through these inducing variables. Therefore, the inducing inputs have to be chosen in such a way that the covariance information over the entire system is maintained as well as possible. We can do this by including these in the hyperparameters, and optimize in terms of the negative log-likelihood. This approach is applicable in many settings. However, in settings where accurate predictions are necessary, such as generative models (Wang et al., 2008), we cannot use this method.

One other improvement that can be made, is when we are dealing with a Toeplitz PSD matrix, which can be inverted in $O(n^2)$ (Mukherjee and Maiti, 1988). In our GP setting, we can obtain one by applying a stationary (i.e., invariant over the input space) PSD kernel, such as the k_{SE} kernel, over equidistant input samples. However, such a setting is in general not always possible, or at the most inflexible.

Another improvement can be made when we are operating in an on-line modeling environment, in which samples are sequentially to the data set. For this, we consider updating the Cholesky factor U of a PSD matrix K, rather than the matrix itself, when encountering adding a new sample (Seeger, 2008).

Assume we want to add an additional input sample to the $n \times n$-matrix K_{11} with Cholesky decomposition $K_{11} = U_{11}^T U_{11}$. Then we have

$$\begin{bmatrix} U_{11}^T & \mathbf{0} \\ \mathbf{u_{12}}^T & u_{22} \end{bmatrix} \begin{bmatrix} U_{11} & \mathbf{u_{12}} \\ \mathbf{0} & u_{22} \end{bmatrix} = \begin{bmatrix} U_{11}^T U_{11} & U_{11}^T \mathbf{u_{12}} \\ \mathbf{u_{12}}^T U_{11} & \mathbf{u_{12}}^T \mathbf{u_{12}} + u_{22}^2 \end{bmatrix} = \begin{bmatrix} K_{11} & \mathbf{k_{12}} \\ \mathbf{k_{12}}^T & k_{22} \end{bmatrix} \tag{2.15}$$

with unknowns $\mathbf{k_{12}}, \mathbf{u_{12}} \in \mathbb{R}^{n\times 1}$ and $k_{22}, u_{22} \in \mathbb{R}$. We can derive that

- $U_{11}^T \mathbf{u_{12}} = \mathbf{k_{12}}$ (2.16)
- $u_{22} = \sqrt{k_{22} - \mathbf{u_{12}}^T \mathbf{u_{12}}}$ (2.17)

Thus, given our previous Cholesky matrix U_{11}, we can solve the linear system in Equation 2.16 using back substitution in $O(n^2)$ time, and Equation 2.17 in $O(n)$, giving us an approach to update our covariance matrix in quadratic time.

It is also possible to remove a data point in $O(n^2)$ (Seeger, 2008) when old data has become irrelevant or when we have too much data to handle the complexity.

2.4 Multiple Outputs

So far, we only considered processes that model multi-input, single-output functions. For example, the $k_{SE}(\cdot,\cdot)$ kernel in Equation 2.3 can handle inputs with any number of dimensions, but returns only one real number (i.e., the covariance between all dimensions of both inputs).

However, in many situations, we are dealing with multi-dimensional output data. Even though modeling the dimensions as separate GPs is an option, the outputs are often strongly correlated, e.g., when dealing with multiple weather sensors from a local area (Osborne et al., 2008). This valuable information will be lost if one opts for this simplistic approach.

There are various approaches to resolve this issue. One way is to view Gaussian processes as a convolution of Gaussian white noise sources with smoothing kernels (Boyle and Frean, 2005). The basic idea of this method is to produce outputs, each generated from convoluted processes that share a common Gaussian white noise process, which grasps the correlations between both. This approach is demanding in terms of computational cost and requires a lot of storage (Álvarez and Lawrence, 2009). However, various sparse approximations were proposed to counter these issues (Álvarez and Lawrence, 2009; Álvarez et al., 2010).

A more straightforward method considers explicitely incorporating an additional input parameter, namely a label l, to denote the output dimension associated with the input $\mathbf{x}$ and have a cross-dimensional covariance kernel that captures the correlations among the outputs (Osborne et al., 2008; Roberts et al., 2012). Technically, based on this label, we could define different mean and covariance functions for each pair of input labels l, l' and pick the ones appropriate for the associated dimensions. However, it is generally difficult to use separate kernels to construct one valid covariance matrix (Melkumyan and Ramos, 2011). Therefore, we stick to the easier approach of having one shared covariance kernel for all dimensions and adapt it according to the labels provided in the inputs.

First, we transform our single-output covariance kernel into a quasi-equivalent multi-output kernel, i.e., a kernel that provides values for multiple dimensions. We define a label kernel $\widetilde{k}_{label}(\cdot,\cdot)$ as

$$\widetilde{k}_{label}(l, l') = h_l^2 \delta(l, l') \tag{2.18}$$

where l, l' are labels associated with an output dimension, h_l is the output scale for l's dimension and $\delta(\cdot,\cdot)$ the Kronecker delta (i.e., a function that maps to 0 or 1 when both its inputs are respectively not equal or equal). We can then write our wrapper multi-output kernel $\widetilde{k}_m(\cdot,\cdot)$ as

$$\widetilde{k}_m([\mathbf{x}, l], [\mathbf{x'}, l']) = k_s(\mathbf{x}, \mathbf{x'})\widetilde{k}_{label}(l, l') \tag{2.19}$$

where $k_s(\cdot, \cdot)$ is a single-output kernel that considers the inputs $\mathbf{x}, \mathbf{x'}$. Even though at first sight, this kernel seems to capture the same amount of information as multiple single-output kernels. However, Bonilla et al. (2008) prove that incorporating noise on the latent values allows for transfer of covariance information among multiple dimensions when inverting the block-diagonal covariance matrix in Equation 2.6. When the observations are noiseless, the multi-output and single-output approaches are equivalent.

We now introduce a new label kernel $k_{label}(\cdot, \cdot)$ by introducing explicit correlation values, allowing for a more intuitive approach.

$$k_{label}(l, l') = h_l h_{l'} S_{l,l'} \tag{2.20}$$

where h_l, h'_l are output scales for labels l, l' and $S_{l,l'}$ their corresponding value in the correlation matrix S. This matrix is built up from a vector $\boldsymbol{\theta}_\mathbf{c}$ of spherical coordinates (Pinheiro and Bates, 1996), which guarantees a positive definite matrix. The spherical parametrization of S provides a direct interpretation of $\boldsymbol{\theta}_\mathbf{c}$ in terms of correlations and variances among the output dimensions. Moreover, each possible vector $\boldsymbol{\theta}$ maps to a unique matrix S, which allows for a stable optimization of an objective function w.r.t. $\boldsymbol{\theta}_\mathbf{c}$.

The correlation parameters $\boldsymbol{\theta}_\mathbf{c}$ and output scales $\mathbf{h_l}$ for all labels l can be included in our set of hyperparameters $\boldsymbol{\theta}$, such that these can be optimized for our negative log-likelihood function given in Equation 2.10. However, one should be wary of the fact that the size of $\boldsymbol{\theta}_\mathbf{c}$ increases quadratically with the number of output dimensions ($\mathcal{O}(d_{out}^2)$), which can be expensive in terms of time and data to properly optimize these.

Finally, we can define a wrapper multi-output kernel k_m as follows:

$$k_m([\mathbf{x}, l], [\mathbf{x'}, l']) = k_s(\mathbf{x}, \mathbf{x'}) k_{label}(l, l') \tag{2.21}$$

This approach can be used to capture correlations amongst different tasks, i.e., instances of various latent correlated models, as has been done by Bonilla et al. (2008).

Note that for a training set X with n observations, the multi-output kernel provides a matrix $K(X, X)$ of size $\mathcal{O}(d_{out} n)$. Computing its inverse can therefore be computationally expensive, as it requires $O(d_{out}^3 n^3)$ time (see Equations 2.6, 2.10 and 2.11).

2.5 Dynamic Process

GPs are widely used to model spatio-temporal data (Banerjee et al., 2008, 2013) and timeseries (Cunningham et al., 2012; Osborne et al., 2008). A simple and

intuitive way to apply them in such settings, is to define the input locations over the time dimension and model the latent function values as latent states at a certain time frame. Thus, our prior distribution (see Equation 2.4) looks as follows

$$\boldsymbol{\zeta}_{\mathbf{t}} \mid \mathbf{t}, \boldsymbol{\theta} \sim \mathcal{N}(\mathbf{m}(\mathbf{t}), K(\mathbf{t}, \mathbf{t}')) \tag{2.22}$$

where $\mathbf{t}$ is an input vector with time stamps and $\boldsymbol{\zeta}_{\mathbf{t}}$ their corresponding latent states. By observing states $\mathbf{s}_{\mathbf{t}}$ at times $\mathbf{t}$, we can thus built up our posterior distribution (see Equation 2.6) and train our model to predict our mean series.

However, this approach introduces some issues, which originate from the fact that we use absolute time stamps. In contrast to applications such as weather forecasts, many practical settings consider a sequence of observed states over time as training data, but require models that are independent of the actual moment of measurement. For example, human motion consists of a series of pose transitions, that are measured over time, but the actual model of the movement should be independent of when those transitions were executed. Moreover, generalizing over time for dynamic systems such as human motion is difficult, as it is likely that the data set contains movement trajectories with different paces, even though they all refer to the same activity we are trying to model. Both problems are demonstrated in Figure 2.3. We can see here that the series have distinct time scales. Additionally, states measured at the same time stamp have therefore the same input, yet emerged from different latent states.

Thus, exactly for these types of problems, it seems appropriate to introduce a dynamic variant of a time series model. For this, we consider the state space itself as the input domain, rather than the time dimension, as is done by Wang et al. (2008). More specifically, we transfer the sequentiality of events from the time space to the state space, such that our dynamic GP models state transitions instead of independent subsequent frames over time. This means that independently of 'when' a state is measured, it will always map to the same latent state for a fixed length scale. The issue of different scales still exists in this approach. However, given that the sampling rate of state observations is small enough, this should not form too much of a problem, as the observed transitions will be approximately the same. Note that our model should support multiple outputs when it has multiple inputs, since the input and output domain are the same. Even though it is perfectly feasible to include multiple output dimensions (see Section 2.4), we specify the formalisms for single-output GPs for the sake of simplicity and notation.

Our new dynamic GP looks as follows:

$$\zeta_{t+1} \sim \mathcal{GP}(m(s_t), k(s_t, s_{t'})) \tag{2.23}$$

where s_t is an input state at time index t and ζ_{t+1} its associated latent next state.

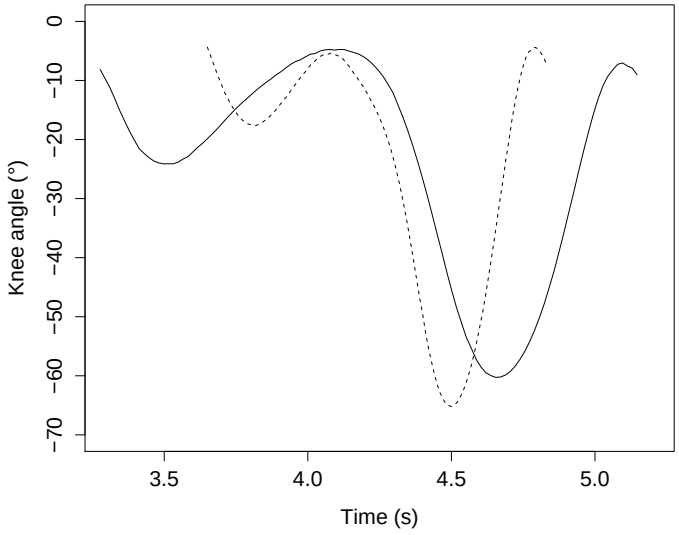


Figure 2.3: Knee angle for one period of a gait cycle, plotted against the time of measurement for two distinct humans walking. One trajectory has a shorter time scale and starts off later (solid), while the other trajectory has a longer time scale and starts off sooner (dashed).

Wang et al. (2008) show that our dynamic GP is actually a Markov chain (MC). These are stochastic processes that model transitions in a system of states (Markov, 1906; Grinstead and Snell, 1988). Innate to these models is the fact that the transition probabilities from one state to another can only depend on a finite sequence of last encountered states. This type of conditioning of a state is called the *Markov property*.

Marginalizing out the parameters $\mathbf{g}$ of the underlying dynamic parametric model from the prior distribution given a subsequent set of input states $\mathbf{s} = [s_1, \cdots, s_{n-1}]$ with latent states $\boldsymbol{\zeta} = [\zeta_2, \cdots, \zeta_n]$, we can induce the Markov property as follows:

$$\begin{aligned} p(\boldsymbol{\zeta} \mid \mathbf{s}, \boldsymbol{\theta}) &= \int p(\boldsymbol{\zeta} \mid \mathbf{s}, \mathbf{g}, \boldsymbol{\theta}) p(\mathbf{s} \mid \mathbf{g}, \boldsymbol{\theta}) p(\mathbf{g} \mid \boldsymbol{\theta}) d\mathbf{g} \\ &= p(s_1) \int \prod_{t=2}^{n} p(\zeta_t \mid s_{t-1}, \mathbf{g}, \boldsymbol{\theta}) p(\mathbf{g} \mid \mathbf{s}, \boldsymbol{\theta}) d\mathbf{g} \end{aligned} \tag{2.24}$$

which considers a first-order MC, as the transition probabilities $p(\zeta_t \mid s_{t-1}, \mathbf{g}, \boldsymbol{\theta})$ are only conditioned on the previous state. Wang et al. (2008) note that putting an isotropic (i.e., uniform in all directions) Gaussian prior on the dynamic parameters $\mathbf{g}$, we obtain

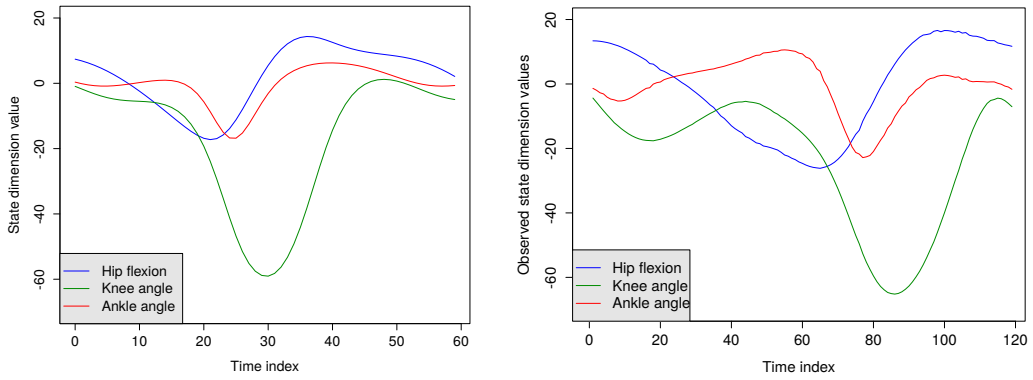
$$p(\boldsymbol{\zeta}|\mathbf{s}, \boldsymbol{\theta}) = \frac{p(s_1)}{\sqrt{(2\pi)^{(n-1)d} |K(\mathbf{s}_{1:n-1}, \mathbf{s}_{1:n-1})|^d}} exp(-\frac{1}{2} tr(K(\mathbf{s}_{1:n-1}, \mathbf{s}_{1:n-1})^{-1} \mathbf{s}_{2:n}, \mathbf{s}_{2:n}^T)) \tag{2.25}$$

which is analogous to the PDF of a Gaussian prior distribution (see Equation 2.5), with the exception of the $p(s_1)$ factor. To have a theoretically sound equivalence between the MC and our GP model, this factor can be removed when the starting state of a batch of subsequent states is always the same.

Note that we can also apply higher-order Markov chains by conditioning on multiple past states. For example, this is useful in practical settings where velocity (second-order) or acceleration (third-order) must be modeled.

An interesting property of our dynamic model is that it is generative. This means we are able to predict a batch of consecutive states given an initial state. We can see this generative aspect applied to human motion in Figures 2.4a and 2.5, which respectively show a two- and three-dimensional representation of a generated gait cycle. Compared to Figure 2.4a, we can see that even though the scales of the trajectories differ, the generated trajectory is similar the observed one. The former is smoother due to the fact it is drawn from a model that is generalized over multiple trials.

This generative property can be useful in computer vision, where a frame-by-frame reconstruction of a walking human can be made (Wang et al., 2008).



(a) Generated (b) Observed

Figure 2.4: 2D representation of generated gait cycle using a dynamic GP model (a) and an actual observed cycle performed by a human (b). A state in the cycle consists of three dimensions, namely the hip flexion (blue), the knee angle (green) and the ankle angle (red).

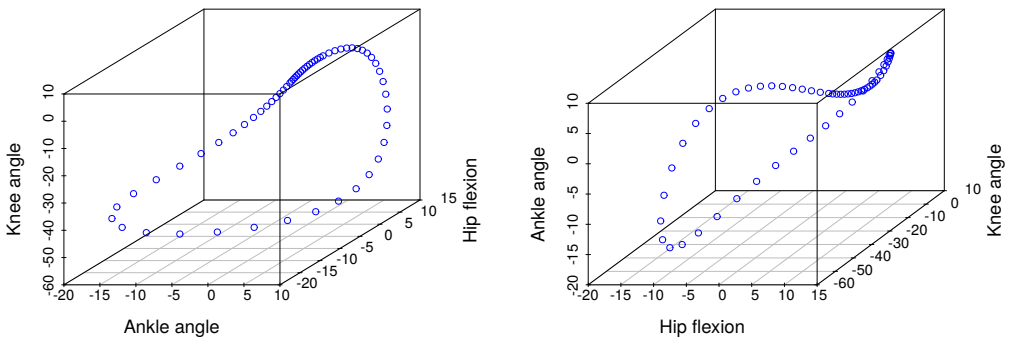


Figure 2.5: 3D representations of generated gait cycle using a dynamic GP model. The right plot is a rotated version of the left one for the sake of clarity.

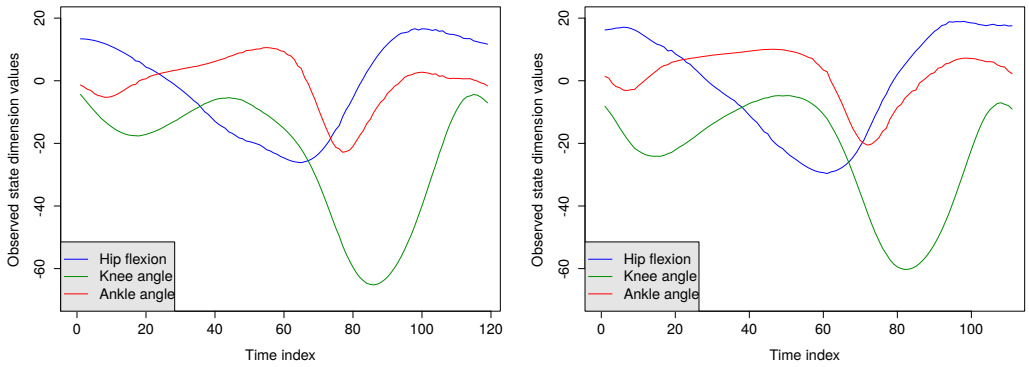
3

MODELS

Modeling human motion is a difficult problem, due to its a high-dimensional state space, and thus, in general, has many degrees of freedom (DoF). To be exact, a human body has 244 DoF (Zatsiorsky and Prilutsky, 2012), for which the hip has three DoF (i.e., flexion, rotation and adduction). However, during simple and general motion tasks, such as walking, humans move in approximately the same way every time. Thus, most of the possible states we encounter during such specific activities are concentrated in a small region of the state space. We can see this in Figure 3.1, where even cross-human trajectories are roughly the same for a gait cycle (i.e., a regular movement pattern during walking). This means that during such motion tasks, for an observed sequence of poses, the dimensions that define these states are highly correlated over the trajectory.

To this end, as we only consider lower-limb activities, we model the right leg, for which its pose is specified by three parameters, namely, one for each joint, i.e., hip flexion, knee angle and ankle angle. Anatomically speaking, the dimensions we use are along the sagittal plane (see Figure 3.2). We can thus reduce this high-dimensional state space to a three dimensional space without losing too much information about the motion itself.

Note that we take physically existing dimensions of a state, rather than reducing the space to a lower dimensional latent space, as is done in most dimension reduction algorithms such as Principal Component Analysis (Smith, 2002) or Gaussian Process Latent Variable Models (Lawrence, 2004, 2005). These algorithms might provide more accurate results, since they compress information about the actual state space in a more advanced way, opposed to just pruning re-



(a) Subject 1 (b) Subject 2

Figure 3.1: Gait cycles performed by two distinct human subjects, for which the state transitions in terms of hip, knee and ankle parameters are recorded. The poses are measured every 0.01 seconds

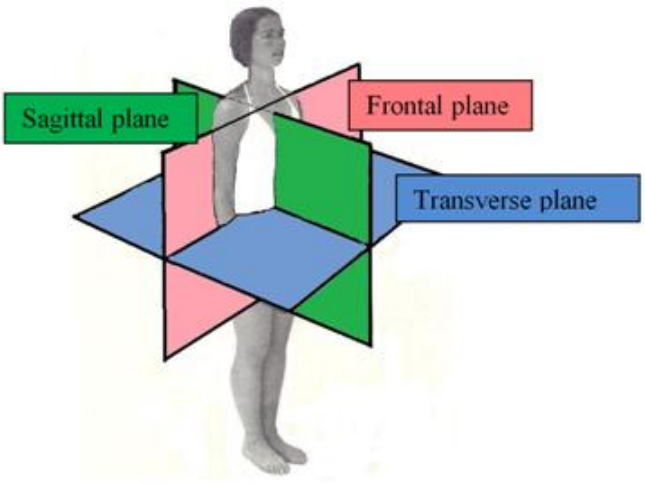


Figure 3.2: Representation of the anatomical planes, i.e., the sagittal, frontal and transverse plane.

dundant dimensions as we do. However, working with the sagittal plane is easier in a practical setting, as they are directly available to the exoskeleton in an on-line environment.

We now formalize the types of models and trajectories we consider in our exoskeleton-assisted human motion setting. First, we define normal and weak models. The former serves as a baseline for a specific task requirement, which we use as a generator for "optimal" trajectories w.r.t. that task. The latter are human-specific processes that models trajectories with certain capability gaps, i.e., deviations from the normal model. Next, we formalize this deviation as a difference between the normal and weak processes. Finally, we can define a support model, which accounts for that deviation and transforms a weak trajectory into a normal one.

The approach of maintaining human-specific deviation models rather than modeling their 'deviated' trajectories has the advantage that it readily implies the needed support. More importantly, such a person-specific support model can be learned on all tasks jointly, and applied to new unseen tasks, as long as a normal model of them is available. A multi-task support model makes sense, as the deviations are likely to be related across tasks, and re-learning them from scratch every time introduces redundancy.

Remember that we can define a model as either a time-series model (i.e., a process mapping absolute time stamps to latent states) or a dynamic model (i.e., a process mapping observed states to their next latent state). As explained earlier in Section 2.5, the latter gives us a more flexible way to shift the model in time. We define our models to be dynamic multi-output GPs, mapping from an observed to a latent state.

3.1 Trajectories

3.1.1 Normal

People with a regular movement pattern require no support from our exoskeleton. Although individually, or even collectively, we cannot grasp the concept of a 'perfect' motion trajectory, we consider a model based on these patterns to be an example of optimal physical behavior.

We define the random process that captures this behavior to be the *normal* process $\eta^a(\cdot)$, where a is the activity we are trying to model.[1] An instance of that process is called the normal trajectory. We approximate such an instance by considering a finite sequence of poses S_t^η at time indices t. The latent function

[1]For the sake of future ease, we omit the index a whenever the context is clear or when the activity is irrelevant.

values associated with these are defined as $\boldsymbol{\zeta^{\eta}_{t+1}}$. From now on, we do not make the distinction between a trajectory and its approximations anymore.

We can define $\eta(\cdot)$ as follows:

$$\eta(\mathbf{s_t}) \sim \mathcal{GP}(m_0, k^{\eta}(\mathbf{s_t}, \mathbf{s'_t})) \tag{3.1}$$

with prior distribution

$$\boldsymbol{\zeta^{\eta}_{t+1}} \sim \mathcal{GP}(\mathbf{0}, K^{\eta}(S^{\eta}_{\mathbf{t}}, S^{\eta}_{\mathbf{t}})) \tag{3.2}$$

Note that we defined $\eta^a(\cdot)$ to be an optimal trajectory model, thus is considered fixed for all humans. This means that for a specific task a, the model can be learned off-line and does not have to be altered afterwards. We can learn this model by generalizing over multiple independent trials of a certain activity.

3.1.2 Weak

Paraplegics (i.e., people whom we aim to help by providing leg support) do not have a normal trajectory as we have defined before. Instead, their poses are drawn from a process that deviates from it. For a specific person p, we refer to this as a weak process $\omega_p(\mathbf{s_t}) = \eta(\mathbf{s_t}) - \widetilde{\delta}_p(\mathbf{s_t})$ with non-zero some *deviation* process $\widetilde{\delta}_p(\cdot)$. We call its observed sequence of weak poses $S^{\omega_p}_{\mathbf{t}}$ with corresponding latent function values $\boldsymbol{\zeta^{\omega_p}_{t+1}}$.

Formally, we have

$$\omega_p(\mathbf{s_t}) \sim \mathcal{GP}(m_0, (\mathbf{s_t}), k^{\omega_p}(\mathbf{s_t}, \mathbf{s_{t'}})) \tag{3.3}$$

with prior distribution

$$\boldsymbol{\zeta^{\omega_p}_{t+1}} \sim \mathcal{GP}(\mathbf{0}, K^{\omega_p}(S^{\omega_p}_{\mathbf{t}}, S^{\omega_p}_{\mathbf{t}})) \tag{3.4}$$

Note that, as opposed to our normal model, $\omega_p(\cdot)$ is human-specific (as is $\widetilde{\delta}_p(\cdot)$). This is mandatory, as we want to provide support to counter a specific disability, which is different per person. Moreover, as the weak trajectories are the ones we observe in our exoskeleton-assisted setting, we have to adjust them in an on-line manner in order to provide real-time support.

3.2 Support

In the previous section, we defined $\widetilde{\delta}_p(\mathbf{s_t}) = \eta(\mathbf{s_t}) - \omega_p(\mathbf{s_t})$ to be the deviation from the normal process for a specific person p. A more practical way to formalize

the relationship between both motion models, is to consider a *reduction* process $\delta_p(\cdot)$, which maps an observed normal state to a latent weak state. Our GP looks as follows:

$$\delta_p(\mathbf{s_t}) \sim \mathcal{GP}(m_0, k^{\delta^p}(\mathbf{s_t}, \mathbf{s_{t'}})) \tag{3.5}$$

with prior distribution

$$\widehat{\boldsymbol{\zeta}}_{\mathbf{t}}^{\boldsymbol{\omega}_\mathbf{p}} \sim \mathcal{GP}(\mathbf{0}, K^{\delta_p}(S_{\mathbf{t}}^{\eta}, S_{\mathbf{t}}^{\eta})) \tag{3.6}$$

where $\widehat{\boldsymbol{\zeta}}_{\mathbf{t}}^{\boldsymbol{\omega}_\mathbf{p}}$ models the same state as $\boldsymbol{\zeta}_{\mathbf{t}}^{\boldsymbol{\omega}_\mathbf{p}}$, but refers to a different random variable. We can now model 'weakness' in the trajectories by applying the reduction process to a sequence of normal states, to get their corresponding weak states.

However, it is not the reduction or deviation we are particularly interested in. We need to define a model that can provide support to weak states in order to transform them into their normal counterpart. Thus, we define a *support* process $\delta_p^{-1}(\cdot)$ that maps an observed weak state to a normal latent state. In some sense, the required support can be seen as the inverse of the reduction, given by $\delta_p(\cdot)$.

We formalize our support process as follows:

$$\delta_p^{-1}(\mathbf{s_t}) \sim \mathcal{GP}(m_0, k_{\delta_p^{-1}}(\mathbf{s_t}, \mathbf{s_{t'}})) \tag{3.7}$$

with prior distribution

$$\widehat{\boldsymbol{\zeta}}_{\mathbf{t}}^{\eta} \sim \mathcal{GP}(\mathbf{0}, K^{\delta_p^{-1}}(S_{\mathbf{t}}^{\omega_p}, S_{\mathbf{t}}^{\omega_p})) \tag{3.8}$$

where $\widehat{\boldsymbol{\zeta}}_{\mathbf{t}}^{\eta}$ models the same state as $\boldsymbol{\zeta}_{\mathbf{t}}^{\eta}$, but refers to a different random variable.

Using this model, we can provide on-line support by observing weak states and map them to their corresponding predicted normal state.

A diagram of relationships between all states in terms of the defined models is Figure 3.3.

3.3 Scale-Dependent Transitions

Observing the way humans move, it is apparent that they perform certain motion activities at different paces. Even when we focus on one specific human, walking will likely be done at different velocities. Therefore, it is crucial to the learning of such models that they can handle changes in velocity.

We define our (time-)scale-dependent process the same way as we have done previously for the scale-independent cases. The variation to it is that we introduce an additional scale input parameter, and alter the training data such that it covers multiple instantiations of this parameter and is able to interpolate to multiple scales.

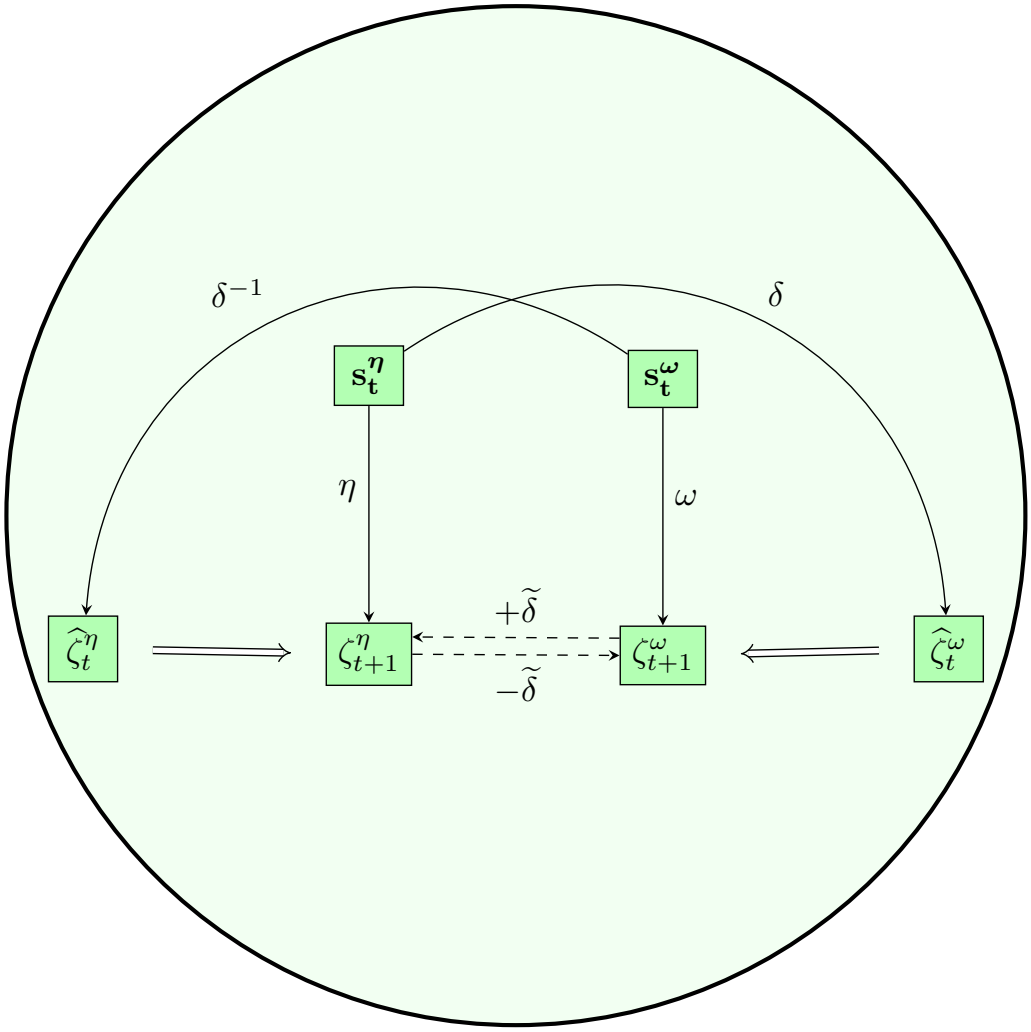


Figure 3.3: Diagram representing the system of models and states. The transitions are either applications of a process on an input state $\mathbf{s_t}$ to achieve a latent state ζ (solid), additions/subtractions of deviation processes $\widetilde{\gamma}(\cdot)$ (dashed) or state transitions (double). The normal and weak processes $\eta(\cdot)$,$\omega(\cdot)$ respectively map an observed normal and weak state $\mathbf{s_t^n}$,$\mathbf{s_t^{\omega}}$ to the next latent normal and weak state ζ_{t+1}^{η},ζ_{t+1}^{ω}. The reduction process $\delta(\cdot)$ maps a normal observed state s_t^{η} to a weak latent state $\widehat{\zeta}_t^{\omega}$. In contrast, the support model $\delta^{-1}(\cdot)$ maps a weak observed $\mathbf{s_t^{\omega}}$ to a normal latent state $\widehat{\zeta}_t^{\eta}$. Note that $\widehat{\zeta}_t^{\eta}$ is a different random variable than ζ_t^{η}, yet are referring to the same underlying state at time t. The sequence of events to learn support is to align a weak trajectory with a normal trajectory, by observing a sequence of poses $\mathbf{s_t^{\omega}}$ and generating a corresponding normal state $\mathbf{s_t^{\eta}}$ using $\eta(\cdot)$, after which $\delta^{-1}(\cdot)$ can be updated. To provide support, we observe a weak state $\mathbf{s_t^{\omega}}$ and apply the support model $\delta^{-1}(\cdot)$ to obtain $\widehat{\zeta}_t^{\eta}$. Its mean could then be used as a predicted state s_t^{η}.

Formally, we have

$$\gamma([r_t, \mathbf{s_t}]) \sim \mathcal{GP}(m_0, k^\gamma([r_t, \mathbf{s_t}], [r'_t, \mathbf{s'_t}])) \tag{3.9}$$

for a scale-dependent process $\gamma(\cdot)$. We assume from now on that every model is scale-dependent, unless specified otherwise.

3.3.1 Velocity

Given the states of a trajectory, we are able to approximate the velocity at a given state $\mathbf{s_t}$.

$$\mathbf{v_t^\gamma} = \mathbf{s_{t+1}^\gamma} - \mathbf{s_t^\gamma} \tag{3.10}$$

where r_t is an observed scale and $\mathbf{v_t^\gamma}$ is the velocity vector at observed state $\mathbf{s_t^\gamma}$, sampled from the process γ.

Note that this velocity can also be modeled over the state space using a GP, by providing a mean and covariance function that maps state input to random velocity variables. Modeling a velocity process instead of a state one provides smoother human motion trajectories, as the process generalizes over these less stable second-order functions. Even though it is not useful anymore to model higher-order functions, as the means of all random variables are close to zero in this case, it is possible to do so next to the velocity model, such that we get even smoother trajectories.

However, it is not necessarily smoothness we want, as generalizing over velocity vectors might give us too much of an approximation to provide proper support. Therefore, we consider it is sufficient to apply Equation 3.10 when we require this second-order information.

3.3.2 Data Augmentation

As mentioned before, we introduce a new scale input parameter. This scale is a property of the velocity vector between the input state, and the next state returned by the process. More specifically, given a certain trajectory and an observed state, we specify an additional velocity scale in order to describe the "units of time" that should pass between this state and the next one. This way, we can handle multiple paces and a certain state, and therefore model changes in paces.

However, in order to train such a model, we need trajectories with their associated artificial scale a priori. This is a difficult task, as the scales should be correctly specified relative to one another (e.g., the scale of a trajectory that has been executed twice as fast should have twice the scale).

We solve this problem by altering our data set. First, we assume that the trajectories in our data set are already normalized. This gives us equally scaled

trajectories to begin with, which makes the task of training this model much simpler. Given this premise, we assign one "unit of scale" to these trajectories. Next, we rescale our trajectories by certain factors (by resampling the data according to the desired scale) and assign these factors to the scale parameters of their states.[2] This allows us to train the model over different scales, such that it can interpolate (and extrapolate) towards unknown ones. After training, we have a model to which we can provide a scale parameter and input state, and retrieve the next state according to the specified scaling of the trajectory. These constraints can be applied in practice by requiring the human subjects (from which we collect the data) to walk at different paces (e.g., by using a treadmill) and use the speed as an observed scale over the entire trajectory.

Note that the assumption about having normalized trajectories in our data set is in practice not true. However, if the *observed* scales $\mathbf{r_t}$ of the trajectories are not too varying, the process should be able to generalize these trajectories to the same *model* scales $\boldsymbol{\rho}_\mathbf{t}$, and do the same for their manipulated variants.

[2]This also works for real factors, in contrast to integer ones. However, this requires applying other more complex scaling techniques on the training trajectories, which we do not consider.

4

LEARNING

Our objective is to provide support to people with weak trajectories, such that they are transformed into normal trajectories. To do so, we observe their weak trajectory state by state and use the models specified in Section 3. A scale-dependent normal model to represent the regular gaits can be learnt off-line using multiple trajectories from various subjects.

However, training a support model requires an alignment of the observed weak trajectory with its associated normal trajectory in order to provide a mapping between both. We provide a novel on-line learning strategy, called the Scale Extraction and Adaptation Method (SEAM).

We discuss the steps that make up SEAM. First, we explain how we can estimate an approximation to the normal model scale $\widehat{r}_t^{\eta}$ from an observed weak vector v_t^{ω}. Then we introduce a confidence measure to denote the certainty of our extracted scale. Using a sequence of scales and confidence, we propose a running weighted average window to aggregate these to obtain the final predicted normal scale. We then discuss the alignment procedure and the limitations of SEAM. Afterwards, we explain how support can be learned in both a single-task and multi-task setting.

4.1 Scale Extraction and Adaptation Method

Even though we provided scales for each trajectory in the training data for a scale-dependent model, it is hard and counter-intuitive to specify a scale to retrieve the

appropriate predicted state. Moreover, given that we need to be able to predict a normal state from the 'associated' weak state, it is required to extract the scale information for the normal trajectory from the weak trajectory. We propose a scale extraction method to retrieve the scale information on-line from a weak velocity vector and apply it our normal model in order to find the associated predicted normal state. In short, we do this aligning the observed weak trajectory with its predicted normal trajectory.

We start off having our observed vector of weak states $S^{\omega_p} = [\mathbf{s_1^{\omega_p}}, \cdots, \mathbf{s_t^{\omega_p}}]^T$ already mapped to their corresponding normal states $S^{\eta} = [\mathbf{s_1^{\eta}}, \cdots, \mathbf{s_t^{\eta}}]$. Observing weak state $\mathbf{s_{t+1}^{\omega_p}}$, we can compute the velocity $\mathbf{v_t^{\omega_p}}$ at $\mathbf{s_t^{\omega_p}}$ by using Equation 3.10.

Given this weak velocity vector, we must extract the scale information w.r.t. the weak latent trajectory and transform it accordingly into scale information w.r.t. to the normal latent trajectory. However, as the model scales ρ_t^{η} are generally unknown for a desired state transition, and the observed scale $r_t^{\omega_p}$ originates from a different process, it is hard to infer what scale r_t^{η} we should provide to our model to get the wanted next predicted normal state.

4.1.1 Separating Scale from Deviation Information

The proposed method for inferring the correct model scale ρ_t^{η} originates from the orthogonal relationship between the observed weak velocity vector $\mathbf{v_t^{\omega_p}}$ with corresponding scale $r_t^{\omega_p}$ and a predicted normal velocity vector $\mathbf{v_t^{\eta}}$ with associated scale $r_t^{\eta} = \widehat{\rho}_t^{\eta}$.

More specifically, the discrepancies in time scales can be seen as 'horizontal difference' Δ_t^h, since a velocity vector describes the displacement over time, while its scale characterizes its instantiation of time. Formally, we have

$$\Delta_t^h = (\mathrm{T}_{t+1}^{\eta} - \mathrm{T}_t^{\eta}) - (\mathrm{T}_{t+1}^{\omega_p} - \mathrm{T}_t^{\omega_p}) \tag{4.1}$$

where $\mathrm{T}^{\gamma}{}_t$ is the time of measurement for state $\mathbf{s_t^{\gamma}}$

The difference between the means of two corresponding latent states $\boldsymbol{\zeta}_{\mathbf{t+1}}^{\boldsymbol{\omega_p}}$ and $\boldsymbol{\zeta}_{\mathbf{t+1}}^{\boldsymbol{\eta}}$ (of respectively the last observed weak state $\mathbf{s_t^{\omega_p}}$ and its predicted normal state $\mathbf{s_t^{\eta}}$) can be viewed as 'vertical difference' Δ_t^v. More formally, given the model scale ρ_t^{η} that provides us with the appropriate latent state $\boldsymbol{\zeta}_{\mathbf{t+1}}^{\boldsymbol{\eta}}$, we have

$$\Delta_t^v = \boldsymbol{\zeta}_{\mathbf{t+1}}^{\boldsymbol{\eta}} - \boldsymbol{\zeta}_{\mathbf{t+1}}^{\boldsymbol{\omega_p}} \tag{4.2}$$

As the dimensions over which these two differences operate, are orthogonal, we could easily separate both information pieces by calculating the observed horizontal difference and then infer the vertical difference using the observed velocity vector.

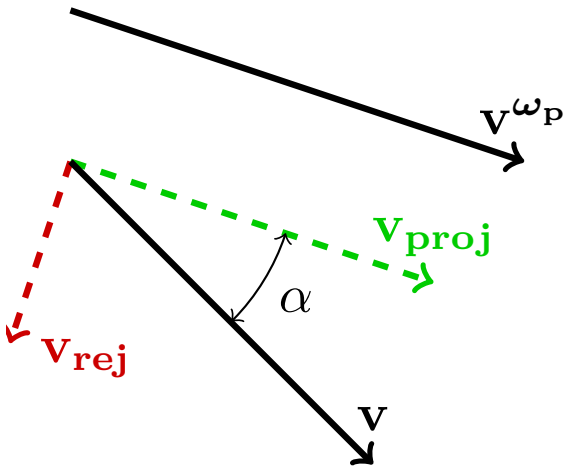


Figure 4.1: Projection and rejection of vector $\mathbf{v}$ w.r.t. vector $\mathbf{v}^{\omega_p}$.

However, we are not dealing with a time-dependent GP, as we are modeling state transitions rather than incorporating time in our input. Therefore, we cannot use exhibitions of time directly to retrieve the difference in scales. We are dealing with this issue by using the axis defined by the weak velocity vector as an analogue to the time axis. This assumption provides us with an orthogonal relationship only defined in terms of our state space, namely the dimension parallel to the weak velocity vector, and the one orthogonal to it.

Consider the normalized weak velocity vector $\hat{\mathbf{v}}^{\omega_p}$ which defines the horizontal axis, and a velocity vector $\mathbf{v}$ from which we attempt to extract the scale. We can project $\mathbf{v}$ onto the horizontal dimension to retrieve the vector $\mathbf{v_p}$.

$$\begin{aligned} \mathbf{v_{proj}} &= (||\mathbf{v}||cos(\alpha))\hat{\mathbf{v}}^{\omega_p} \\ &= (\mathbf{v}.\hat{\mathbf{v}}^{\omega_p})\hat{\mathbf{v}}^{\omega_p} \end{aligned} \tag{4.3}$$

where α is the angle between $\hat{\mathbf{v}}^{\omega_p}$ and $\mathbf{v}$. We compute the rejected vector as well.

$$\mathbf{v_{rej}} = \mathbf{v} - \mathbf{v_{proj}} \tag{4.4}$$

The obtained vectors can be viewed in Figure 4.1.

Applied to our case, we consider $\mathbf{v}$ to be equal to an approximated normal vector $\widehat{\mathbf{v}}^{\eta}$, obtained using Equation 3.10. Given that we know the observed scale r^{ω_p} of the normal vector and that the considered transition starts at the associated latent states for both processes ω_p and η, we can update the approximated predicted scale $\widehat{r}^{\eta}$ as follows:

$$\widehat{r}^{\eta} = \frac{||\mathbf{v}^{\omega_p}||}{||\mathbf{v}^{\eta}_{\mathbf{proj}}||}\widehat{r}^{\eta} \tag{4.5}$$

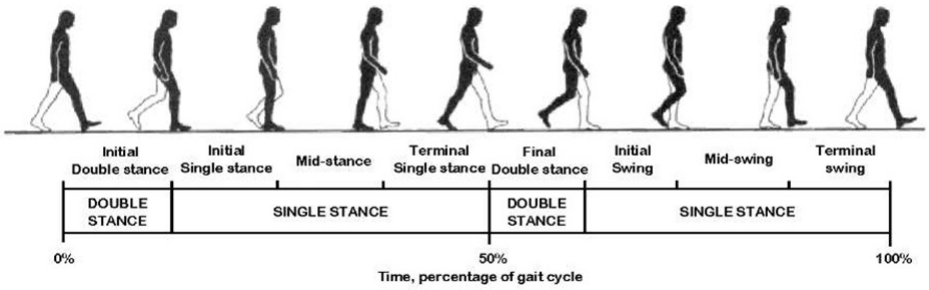


Figure 4.2: Representation showing the different types of poses in a gait cycle. (Martín-F'elez et al., 2011)

Since we are working with orthogonal dimensions, we can extract the vertical difference by applying the same ratio using the rejected vector to the means.

4.1.2 Confident Scale Information

In our case, these transformations are theoretically unsound, as a weak latent state that deviates from the normal process might have the same distribution as another normal latent state further down the road. In other words, there are various solutions to the separation problem we set up, for which our method can assign erroneously information to either the horizontal scale property or the vertical mean property.

Therefore, we consider a more stable approach, in which we introduce a new assumption about the observed weak trajectories. Namely, we assume that the "weakness" segment (i.e., a significant deviation from the normal model) in an observed trajectory occurs in a concentrated region after a considered amount of observations. This allows us to learn a confident scaling of the trajectory beforehand, without worrying too much about it when encountering the weak segment. Moreover, this assumption is especially helpful in settings where the scaling over the entire observed trajectory is approximately the same (w.r.t., the model scale).

Note that this assumption is not unreasonable in most practical cases, as an activity is initialized from a resting position, in which both legs provide support to the entire body. Moreover, in Figure 4.2), we can see there is a segment where the human does not support himself well, namely at the moment the human switches his pivoting foot. This in general the point where the weakness in the cycle will occur.

Using this assumption, we attempt to evaluate how confident we are in the scale information we extracted at a particular weak state. We do this by weighting the information with the inverse squared norm of the rejected normal velocity

vector.

$$conf(\widehat{r}^{\eta}) = \frac{1}{||\mathbf{v}^{\boldsymbol{\eta}}_{\mathbf{rej}}||^2} \tag{4.6}$$

When the direction of $\mathbf{v}^{\boldsymbol{\eta}}$ is mainly in the direction of the corresponding weak velocity vector $\mathbf{v}^{\omega_{\mathrm{P}}}$, we have good reason to believe that there is not a lot of vertical difference that pollutes the scale information. This means we could apply Equation 4.5 without too much trouble. However, when the rejected vector is large, the weak trajectory deviates heavily from the normal model, in which case we have vertical and horizontal information contained in the weak velocity vector.

Note that even when the weak and normal vectors are mainly overlapping, there can still be multiple cases to the separation problem. However, in this case, other solutions than the trivial one (which is that the weak and normal states are considering the closely related latent states) are only appearing when there are a lot of states on the weak trajectory similar to the normal state we are considering. Given a small sampling rate, this phenomenon should not have too much of an impact on the extraction method, which means the trivial solution is the most convincing in our case.

4.1.3 Running Average over Scales

Now that we have a confidence for the predicted scale $\widehat{r}^{\eta}$, we know which information is valuable and which is not. Given our assumption that most of the earlier states in the weak and normal trajectory are quasi-overlapping, we can discard the information we are uncertain about during the weak segment, and only keep the observed scales for which the transitions are much alike over both trajectories. The questions of how we decide whether we should keep information and how we should extrapolate the scales in those sparse uninformative regions, remain.

We can solve both questions by using a moving weighted average over the last n predicted scales $\widehat{\mathbf{r}}^{\boldsymbol{\eta}} = [\widehat{\mathbf{r}}^{\boldsymbol{\eta}}{}_1, \cdots, \widehat{\mathbf{r}}^{\boldsymbol{\eta}}{}_n]^T$ (including the extracted new one $\widehat{\mathbf{r}}^{\boldsymbol{\eta}}{}_n$), where each scale is weighted by their confidence, and use this result as the next scale. More formally, we define the weighted average for our new scale r^{η}_n as follows:

$$r^{\eta}_n = \frac{\sum_{t=1}^{n} conf(\widehat{\mathbf{r}}^{\boldsymbol{\eta}}{}_t)\widehat{\mathbf{r}}^{\boldsymbol{\eta}}{}_t}{\sum_{t=1}^{n} conf(\widehat{\mathbf{r}}^{\boldsymbol{\eta}}{}_t)} \tag{4.7}$$

The reasoning behind introducing such a window, is that significant changes in pace do not occur that often. Thus, extrapolating in this manner gives us an aggregation of the most relevant scales. Additionally, we discard scales with a

confidence below a certain threshold $thresh_{conf}$, in order to not pollute the window with irrelevant scales.

4.1.4 Alignment

Now we are ready to align our trajectories. Assuming we have our states S^{ω_p} already mapped to their associated normal states S^{η}, we can observe the next weak state and compute our velocity vector $v_t^{\omega_p}$. Using the steps defined previously, we can compute an approximate scale from the projection of the predicted normal vector on the observed weak vector. However, we do not have the normal velocity vector available yet, since the next normal state is what have to predict.

To solve this problem, we use an approach much like Expectation-Maximization in which we iteratively try to estimate the required normal vector more accurately. More specifically, we initialize the normal vector by using our moving average to get an estimated scale and use it to predict the next normal state with our scale-dependent normal model $\eta(\cdot)$. Once we have this vector, we can extract the scale ratio and adjust our previous guess to get a better estimate for our normal vector. Even though multiple iterations of this are possible, we found that one is sufficient, and that more iterations do not improve much on the estimated scale.

To summarize, first SEAM attempts to extract the scale ratio between an observed weak velocity vector and a predicted normal vector projected onto the observed one. Updating the current scale of the predicted normal vector with the obtained ratio, we can get a better estimate for this scale. We describe the confidence of an estimate scale as the inverse of the norm of the rejected vector squared. Using both the obtained scale and its confidence, we can add it to a running weighted average window to obtain the final scale to estimate the normal vector. Given our estimated normal vector, we can calculate the new normal state and reiterate the entire process for the next weak states.

4.1.5 Assumptions

Our method relies on three large assumptions, additional to the ones imposed by using a scale-dependent GP.

1. We consider the normal model $\eta(\cdot)$ to be accurate at all times. This means that given a certain scale and state as input, we should assume the predicted state is exactly the one we want. Even though this is probably not true, it is not an unreasonable assumption to make, as we train our normal process on multiple subjects and therefore have a generalized model that captures regular movement patterns.

2. In order to start the alignment of both trajectories, we require the first observed weak state to be already mapped to its corresponding normal state. This is necessary, as our generative GP normal model is not stable when initialization is off. Additionally, the on-line extraction method does not provide guaranteed convergence, which requires a good initialization to keep the trajectories aligned.

3. We should assume that the earlier latent state pairs over both trajectories are close to each other, as we previously mentioned.

All these assumptions are introduced to prevent the premise that trajectories are at all times equally scaled. Compared to this, the three assumptions allow for a looser-constrained domain of application.

4.2 Support

While applying our scale extraction method, we can learn our support model $\delta_p^{-1}(\cdot)$ on-line as well. Every time we observe a weak state and predict its corresponding normal state, we can add this mapping to the support model as training data. Eventually, after various trajectories of the weak subject, we have a model that captures the support needed to account for the subject's disability.

Moreover, once we properly learnt the support, we can apply this model as well while performing the scale extraction method. More specifically, when we observe a weak state, we can use the support model to map it to a corresponding normal state, and extract the scale from that state instead. This way, we have more accurate predictions, as we are dealing with a trajectory that is allegedly closer to the normal trajectory we want to predict. We can then add the achieved mapping again to the support model in order to provide more accurate support. This iterative process should provide on-line support while keeping it up-to-date to the status of the disability.

4.2.1 Multi-Task

As mentioned in Section 2.4, we can capture correlations amongst multiple dimensions by incorporating a dimension label along with the input state, and induce a covariance kernel that models the covariance between the cross-dimensional inputs (see Equation 2.21). This should provide more accurate results when the dimensions are indeed highly correlated, as relevant information is transferred between these dimensions, thus reducing the variance on the latent states.

However, we can apply this approach to various tasks as well. Considering that the same type of motor dysfunction causes the capability gaps over multiple

activities, we expect them to be highly correlated. A multi-task support model can capture these correlations and share covariance information across multiple activities. This provides us with a model that is easier to manage and contains less redundant information that having multiple single-task support models. Moreover, these correlations allow the multi-task model to extrapolate to unseen tasks, which is a useful feature when the exoskeleton encounters an activity it was not trained on.

We define a multi-task support model as $\delta_p^{multi^{-1}}(\cdot)$ and introduce the hyper-parameters that specify the output scales and correlations for the activities respectively as $\mathbf{h}_{\mathbf{l}}^{\mathbf{a}}$ and $\boldsymbol{\theta}_{\mathbf{c}}^{\mathbf{a}}$. This approach has also been considered by (Bonilla et al., 2008), in which they try to predict exam scores of students from different schools (i.e., the tasks).

5

EXPERIMENTS

Our aim is to set up an exoskeleton-assistance framework. This requires learning a support model on-line, which should capture the capability gaps in human-specific trajectories. Afterwards, we can use this model to provide the support needed during a motion task. We evaluate the various aspects that build up our framework.

First, we test our novel trajectory alignment method SEAM. It is crucial for our on-line support framework to have a robust approach for aligning an observed trajectory with its corresponding "optimal" trajectory, generated from the normal model $\eta(\cdot)$. Therefore, we run SEAM on observed normal trajectories in order to assess its stability, after which we evaluate its capability to ignore irrelevant observed scales in weakness segments, by applying it to weak synthetic data.

Next, we perform an experiment in which we learn a support model off-line, and apply it to an observed weak trajectory to transform it into its normal counterpart. We use synthetic data, such that we can compare the predicted normal trajectory with the actual one. Afterwards, we apply our support model on-line in our SEAM setting, specified in the previous experiment, in the hope that SEAM has an easier task to predict model scales.

Then, we assess the utility of our multi-task support model by incorporating more activities using a synthetic data set. Given that motor capabilities of the subject should operate the same over all tasks, it is expected that the capability gaps over multiple activities are correlated.

Finally, we test all the components of our framework in a real-world setting. Namely, we apply it to children's gaits, which are considered to be irregular. Even though we are not providing assistance to paraplegics, the gait cycle of a child

is considered irregular. Note that this is just a proof of concept, rather than an actual experiment, as we do not have an actual normal trajectory associated with the irregular gaits to compare our predictions with.

5.1 Evaluation

In order to evaluate the accuracy of predictions performed by our methods, we introduce the squared error (SE) measure, which is equal to $||\mathbf{s_t} - \mathbf{s'_t}||_2^2$ for paired states $\mathbf{s_t}, \mathbf{s'_t}$. We can then aggregate these errors by taking the mean over the SEs of all pairs of observed states, such that we obtain the mean squared error (MSE). Formally,

$$MSE(S_n, S'_n) = \frac{\sum_{t=1}^{n} ||\mathbf{s_t} - \mathbf{s'_t}||_2^2}{n} \tag{5.1}$$

where $S_n = [\mathbf{s_1}, \cdots, \mathbf{s_n}]$ and $S'_n = [\mathbf{s'_1}, \cdots, \mathbf{s'_n}]$ are two aligned sequences of states. We choose this as a measure of performance, as small errors in predictions are not noticeable in practice, while large ones might become uncomfortable for the wearer of an exoskeleton. Note that this measure can be misleading in our case when used in our setting. As the actual normal trajectories are not necessarily captured by our assumed optimal model, the scenario in which the actual trajectory is less desirable than the predicted one, can occur. In this case, the MSE can be high, even though the predicted trajectory seems to exhibit the desired behavior. Therefore, we mainly use this statistic to compare two methods, and use it carefully, along with plotted trials, to evaluate the methods independently as well.

We use k-fold cross validation over the data of k chosen human subjects, in which we assign the data of one subject to one fold. This way, we can do the experiment k times, where each fold is used as the test set once, while the other $(k-1)$ parts are used as training data. This provides us with more test data to infer reliable conclusions about our results. Moreover, making the test sets subject-specific prevents biasing the test set towards subject-dependent information included in the training set.

Finally, we perform statistical analysis to check whether there is a significant difference between the accuracies of two methods. Until now, since we are using GPs, we assumed the latent states follow a Gaussian distribution. Theoretically, this means that the random variable obtained by applying the MSE formula to a pair of latent states follows a χ^2 distribution (Lancaster, 2006), which prevents us from using the normality assumption in statistical tests, such as the Student's t-test. As we have no reason to believe otherwise, we should therefore settle for a Wilcoxon signed-rank test, in which we compare the observed poses pairwise over all trajectories for each method.

5.2 Data

Our data consists of trajectories for various activities, carried out by different human subjects. The activities we consider are walking (i.e., the gait), standing up from a sitting position with the knee at a starting angle of 120 degrees, and ascending stairs. We respectively call these activities 'gait' ($a = 1$), 'sit to stand' ($a = 2$) and 'stair ascent' ($a = 3$). The activities are visually presented in Figure 5.1.

The data sets we use during the experiments are taken from the work by (Afschrift et al., 2014). They are constructed by monitoring trajectories, in which the poses of the subjects are sampled at a time interval of 0.01 seconds. Using inverse kinematics (IK), the joint parameters for a certain pose are obtained. A pose consists of 37 parameters, containing information about the position of the legs, feet, arms, hands, upper body and lower body, determining the pose of a human. As we focus on providing leg support, we stick to the parameters of the right leg. More specifically, we use the three parameters along the sagittal plane, mentioned in Section 3, namely the hip flexion, the knee angle and the ankle angle, which are sufficient enough to define the state of a leg.

We use two data sets throughout the experiments. The first one, called the ADL (Activities of Daily Living) data set, is our principal data set and contains only 'normal' trajectories. This dataset is used to train and test our normal models, and to benchmark our support models and SEAM against. Additionally, the data set provides various synthetic 'weak' data, in which capability gaps in the normal trajectories have been added to simulate impairments. We use the 30% gap model, which is constructed from an musculoskeletal model, capturing various levels of weakness (Afschrift et al., 2014). In this data set, 7 subjects are available, each who have done 3 trials, giving us 21 trajectories.

The second data set contains the gait cycles of children, whose movements are generally irregular. Even though this does not completely overlap with data containing the gait cycles of paraplegics, using this data set can give an indication of how well our framework adjust irregular movement patterns in practice. We consider 3 children, each which have done 5 independent walking cycles, bringing us to 15 "weak" trajectories.

5.3 SEAM Parameters

As SEAM requires parameters $size_W$ and $thresh_{conf}$ (see Section 4.1.3), we perform a small experiment in order to show the influences of the window size and confidence threshold on the extraction method. We train a scale-dependent normal model $\eta(\cdot)$ on 12 trajectories of 4 human subjects (i.e., subjects $1-4$) with an assumed normal trajectory (3×4 trials), and observe 6 normal trajectories

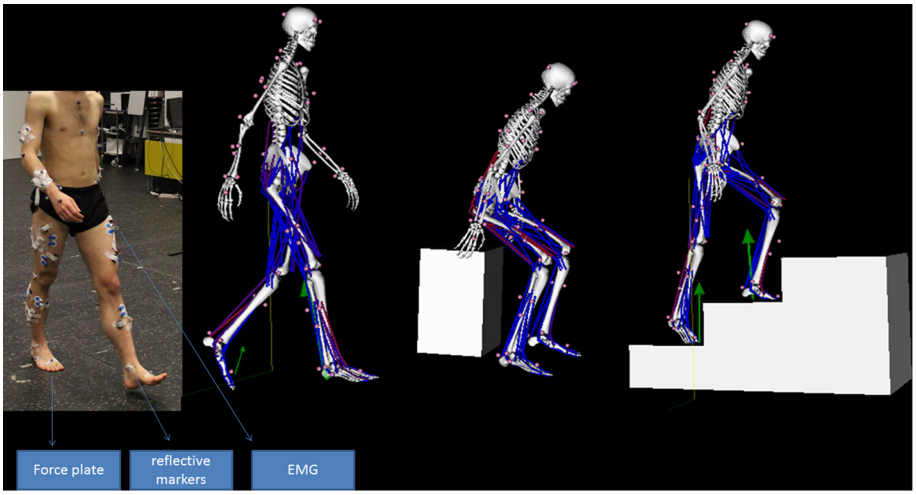


Figure 5.1: The three activities present in the ADL data set. From left to right, we have what we defined as 'gait', 'sit to stand' and 'stair ascent'. (Afschrift et al., 2014)

$size_W$	**subject 6**			**subject 7**		
	trial 1	*trial 2*	*trial 3*	*trial 1*	*trial 2*	*trial 3*
1	1505.56	1397.53	404.59	1773.02	1377.03	1705.53
3	294.75	1480.63	159.93	1679.26	1377.03	1705.53
5	1808.15	1480.63	159.93	1512.91	1377.03	1705.53
7	1480.63	1480.63	159.93	1212.19	1377.03	1705.53

Table 5.1: MSE over the states per trial per $size_W$ (with $thresh_{conf} = 0.01$)

(independent from the training set) of 2 human subjects (i.e., subjects $6 - 7$), drawn from our principal data set. We run our SEAM algorithm for parameters $size_W \in [1, 3, 5, 7]$ and $thresh_{conf} \in [0.001, 0.005, 0.01]$. In order to avoid a quadratic number of runs, we do not consider them jointly. Instead, we first run trials for $size_W$ with $thresh_{conf} = 0.01$, after which we pick the best $size_W$ to run the $thresh_{conf}$ trials with. Note that we picked a 'normal' trajectory, rather than a weak one, to observe. This is specifically to test the stability of our method for various parameters, as small errors in the approximated scale $\widehat{r^\eta}$ can throw the predicted trajectory off track. The mean squared errors (MSE) for the $size_W$ and $thresh_{conf}$ trials are respectively given in Tables 5.1 and 5.2.

We can see that the most of the values are the same, which indicates that the parameters have a small influence on the algorithm in general. Moreover, the MSE values are pretty high, which indicates that the predicted trajectory goes off track a lot. Looking at Figure 5.2b, we can see this issue arising. However, Figure 5.2a

$thresh_{conf}$	**subject 6**			**subject 7**		
	trial 1	*trial 2*	*trial 3*	*trial 1*	*trial 2*	*trial 3*
0.001	1839.38	1397.16	404.59	1783.61	1705.53	1705.53
0.005	1983.78	1480.94	404.59	1388.29	1377.03	1705.53
0.01	294.75	1480.63	159.93	1679.26	1377.03	1705.53

Table 5.2: MSE over the states per trial per $thresh_{conf}$ (with $size_W = 3$).

shows us an instance in which the predicted trajectory appears to converge, which indicates our method is able to get back on the right track.

As the MSE over all states in all trials per parameter setting is as follows:

$\underline{size_W}$

- $1 \rightarrow 1360.54$
- $3 \rightarrow 1116.19$
- $5 \rightarrow 1340.69$
- $7 \rightarrow 1339.44$

$\underline{thresh_{conf}}$

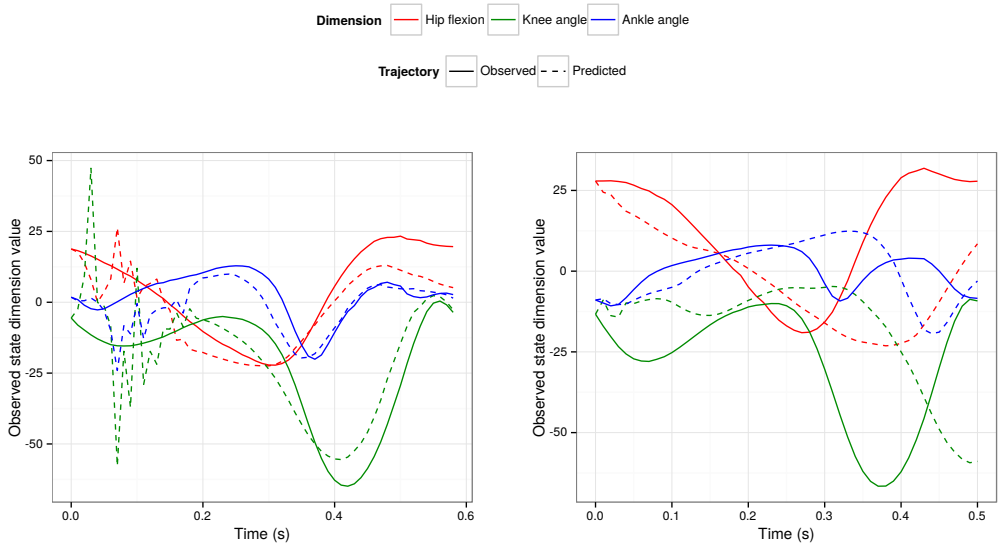
- $0.01 \rightarrow 1081.77$
- $0.005 \rightarrow 1389.63$
- $0.001 \rightarrow 1410.31$

we opt for $size_W = 3$ and $thresh_{conf} = 0.01$ during our experiments in Chapter 5.

Note that we do not use the test set (i.e., the set containing subjects 6 and 7) in our scale-dependent models in our main experiments. However, we do use the set of trajectories on which we trained our normal model, which might introduce a small bias when using these as a test set with the selected parameters later on. In practice, these parameters can be fine-tuned for a specific subject, in which case this small bias is justified.

5.4 SEAM Assessment

This experiment questions the capabilities of our alignment method SEAM. More specifically, we compare the application of SEAM using a scale-dependent normal model $\eta^{scaled}(\cdot)$ with the direct state generation (DSG) by a scale-independent normal model $\eta^{unscaled}(\cdot)$. We first run SEAM and DSG against observed normal trajectories, in order to assess the stability of our algorithms. Afterwards, we run



(a) Converging (b) Diverging

Figure 5.2: SEAM executed on trial 1 of subject 6 (left) and on trial 2 of subject 7 (right), with $size_W = 3$ and $thresh_{conf} = 0.01$.

them against observed weak trajectories to test their applicability in our on-line support framework. For this, we use the ADL 'gait' data set with 5-fold cross-validation over the first 5 subjects, giving us 12 training trajectories and 3 test trials per fold. The weak test trajectories for the latter experiment are synthetically altered versions of the normal test trajectories, as discussed in Section 5.2.

We train the models $\eta^{scaled}(\cdot)$ and $\eta^{unscaled}(\cdot)$ on multiple versions of our available training trajectories. Namely, we resample those by only keeping every β states. This gives us states that are each 0.01β seconds away from their neighbors, keeping them equidistant in time. We train $\eta^{scaled}(\cdot)$ on $\beta = 1$, $\beta = 2$ and $\beta = 3$, while training $\eta^{unscaled}(\cdot)$ on $\beta = 2$. The test trajectories are scaled by a factor of $\beta = 2$.

We use the $k_m(\cdot,\cdot)$ covariance kernel for each of the models, specified in Section 2.4 with $k_s(\cdot,\cdot) = k_{SE}(\cdot,\cdot)$. The hyperparameters are optimized w.r.t. a subset of the training data. To built this set, we randomly select 1 trial per subject, although we make sure the same ones are selected for both models. We apply 100 iterations of conjugate gradient descent for each parameter.

For example, the hyperparameters of $\eta^{scaled}(\cdot)$, trained using the trials of the first 4 subjects, are as follows:

- $\sigma \rightarrow 0.19$
- $\boldsymbol{\lambda} \rightarrow [0.12, 6.26, 9.98, 5.57]$

- $\mathbf{h_l} \rightarrow [1.70, 3.44, 1.75]$
- $\boldsymbol{\theta_c} \rightarrow \begin{bmatrix} 2.68 & -2.28 & -1.19 \\ -2.28 & 3.87 & 1.81 \\ -1.19 & 1.81 & 2.19 \end{bmatrix}$

Note that we do not specify the output scale h used in $k_{SE}(\cdot, \cdot)$. However, as h can entirely be defined by the output scales $\mathbf{h_l}$, we consider it a redundant degree of freedom and decide to omit it.

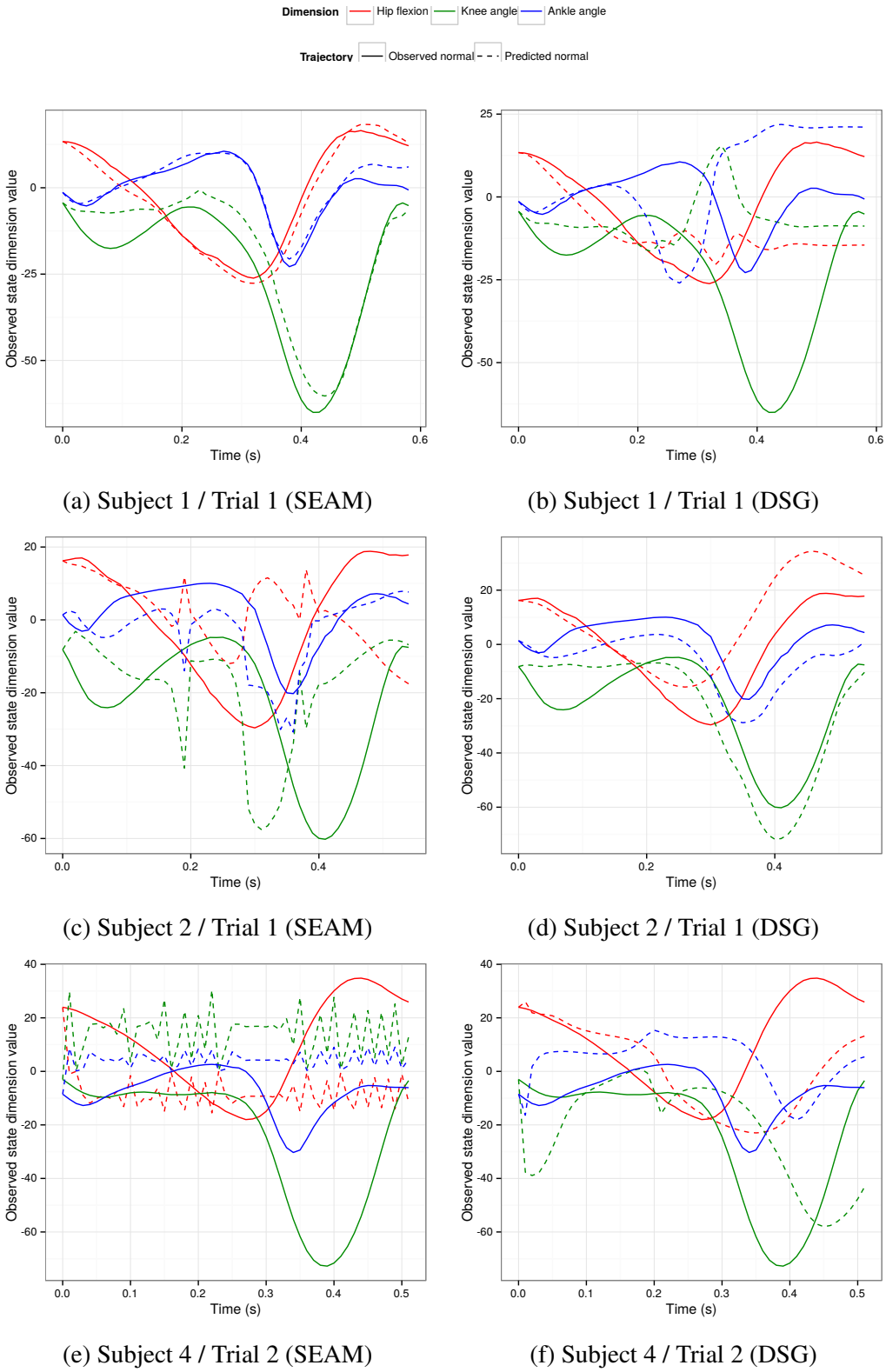
We pick $thresh_{conf} = 0.01$ and $size_W = 3$ as the configuration of SEAM (see Section 5.3). Moreover, we only include an observed scale r_t in our running average window if $0.01 \leq r_t \leq 0.03$, which prevents the method from extrapolating (in contrast to interpolating) to other scales. Even though extrapolation could bring us back on track faster when SEAM deviated from the desired course, we expect SEAM to be an unstable method, given what we have seen in Section 4.1.3. Therefore, this extra constraint provides a more robust setting to work in. Additionally, making fast adjustments in the predicted normal trajectory is unwanted in a practical exoskeleton-support setting, as this is uncomfortable to the wearer of the exoskeleton. To initialize our methods, we do as assumption 2, given in Section 4.1.5, requires and map the initial observed state to its correct normal counterpart. Moreover, as SEAM needs an initial "observed" scale and associated weight as well, we pick respectively 0.02, i.e., the general scaling of the test trajectories, and 1.

5.4.1 Results

Normal Observed Trajectory

When we look at Figure 5.3, we can see 3 trials plotted for each approach against a normal observed trajectory. In plots (a) and (b), the pair of trials for which SEAM performed the best are shown. In this case, SEAM mainly manages to stay close to the observed normal trajectory, while DSG goes off track after 0.2 seconds. Until 0.4 seconds, it still attempts to predict a trajectory, but using the wrong scale. After 0.4 seconds, each dimension gets drawn towards a constant value. In plots (c) and (d), we have the trial pair for which DSG performs the best. It nearly perfectly predicts the desired normal trajectory, while SEAM fails to find a proper scaling. Plots (e) and (f) show us respectively the trial in which SEAM performs the worst and the corresponding DSG trial. SEAM does not manage to form a scaled version of the desired trajectory, while even though the trajectories in the DSG case are not aligned, we can see a scaled version appearing at the end.

Tables 5.3 and 5.4 show respectively the MSE per trajectory for the DSG and SEAM methods. The ones presented in Figure 5.3 are highlighted. Note that one



(a) Subject 1 / Trial 1 (SEAM)

(b) Subject 1 / Trial 1 (DSG)

(c) Subject 2 / Trial 1 (SEAM)

(d) Subject 2 / Trial 1 (DSG)

(e) Subject 4 / Trial 2 (SEAM)

(f) Subject 4 / Trial 2 (DSG)

Figure 5.3: Predictions for SEAM (left) and DSG (right) plotted against the observed normal trajectory. The top and middle two plots respectively present the trial pairs for which SEAM and DSG performs best. The bottom two plots shows the trials for which SEAM is the worst.

DSG	**trial 1**	**trial 2**	**trial 3**
subject 1	1509.83	2078.33	1865.26
subject 2	343.24	310.05	450.84
subject 3	561.68	784.64	422.243
subject 4	1003.85	1245.99	1823.69
subject 5	1187.26	852.48	1501.95

Table 5.3: MSE for each of the 15 runs of DSG, applied to an observed normal trajectory. Each subject has a differently trained $\eta^{unscaled}(\cdot)$. The highlighted entries are shown in Figure 5.3.

SEAM	**trial 1**	**trial 2**	**trial 3**
subject 1	53.43	138.33	1295.49
subject 2	972.56	724.62	1036.61
subject 3	1100.84	1082.40	1176.64
subject 4	2994.45	2999.76	2864.94
subject 5	1933.83	1643.49	1325.91

Table 5.4: MSE for each of the 15 runs of SEAM, applied to an observed normal trajectory. Each subject has a differently trained $\eta^{scaled}(\cdot)$. The highlighted entries are shown in Figure 5.3.

row of trials uses the same model, as is described by our 5-fold cross validation approach. We can see that the MSEs are high in many cases, which indicates the instability of our generative models (as can be observed in Figure 5.3). Additionally, the top two accuracies are due to SEAM (i.e., for trial 1 and 2 of subject 1). However, when we compute the MSE over all observed state pairs over all trajectories, we have 1069.81 for DSG and 1391.18 for SEAM, indicating that DSG performed better in this case.

Yet, the meta-results given by our pair-wise Wilcoxon tests tell us that, with values ($W = 325390, p = 0.051$), the obtained results are insignificant. Therefore, we cannot make any assumptions about the relative performance of both methods.

Weak Observed Trajectory

Figure 5.4 presents SEAM and DSG predictions w.r.t. an observed weak trajectory with a synthetic capability gap. Again, plots (a) and (b) show the trial pairs for which SEAM performs best. It seems to estimate the scale of the trajectory fairly well, while DSG loses his grip after 0.2 seconds. Plots (c) and (d) show the best performing DSG trial and its corresponding SEAM trial. DSG follows the

DSG	**trial 1**	**trial 2**	**trial 3**
subject 1	1509.83	2078.33	1865.26
subject 2	343.24	310.05	450.84
subject 3	561.68	784.64	422.24
subject 4	1003.85	1245.99	1823.69
subject 5	1187.26	852.48	1501.95

Table 5.5: MSE for each of the 15 runs of DSG, applied to an observed weak trajectory. Each subject has a differently trained $\eta^{unscaled}(\cdot)$. The highlighted entries are shown in Figure 5.4.

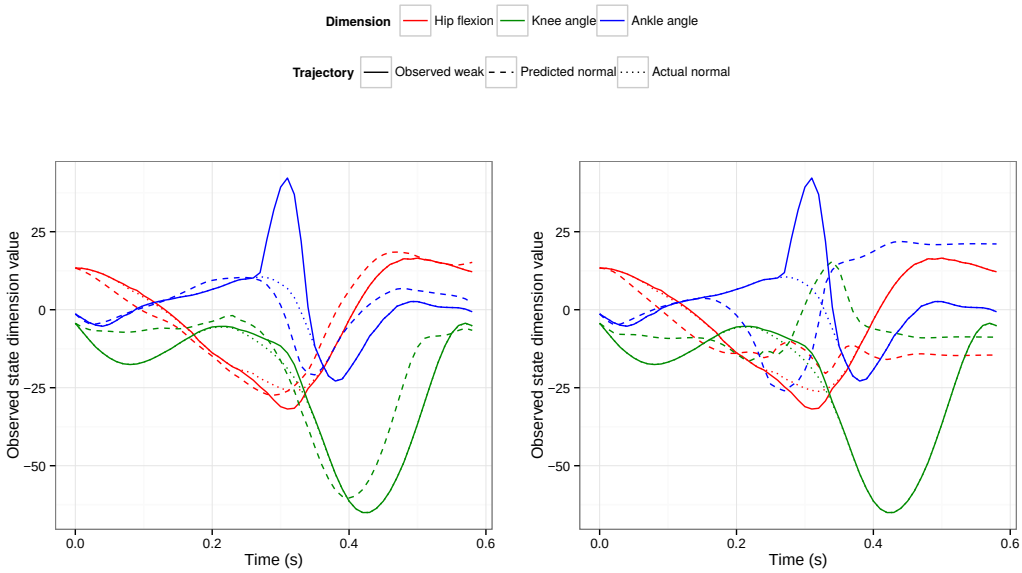
SEAM	**trial 1**	**trial 2**	**trial 3**
subject 1	158.91	167.54	1357.88
subject 2	1470.47	1422.89	1180.39
subject 3	1371.89	1383.74	1469.81
subject 4	2999.21	3014.88	2826.99
subject 5	1758.87	1446.46	1592.55

Table 5.6: MSE for each of the 15 runs of SEAM, applied to an observed weak trajectory. Each subject has a differently trained $\eta^{scaled}(\cdot)$. The highlighted entries are shown in Figure 5.4.

trajectory well, except that the scales around 0.04 seconds are a bit smaller than the desired scale. However, if we look for example at the ankle angle at that point, the observed trajectory makes a sudden turn, while our predicted ankle state smoothly adjusts itself, which is wanted behaviour in a practical setting. Additionally, even when SEAM does not know what to do in the beginning, we can see a trajectory appearing right at the end. The last pair of plots shows us the worst SEAM trial, next to the corresponding DSG one.

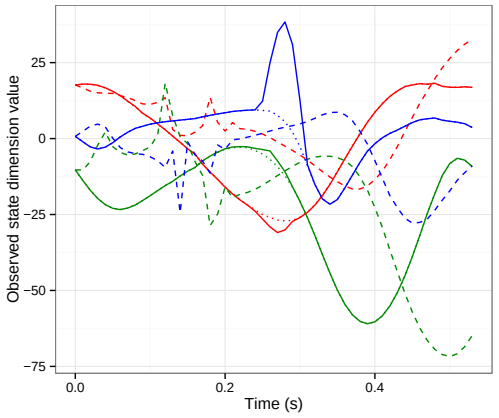
Tables 5.5 and 5.6 show the MSE over all observed state pairs for respectively DSG and SEAM, with the ones in Figure 5.4 highlighted. Now, the MSEs for SEAM are even higher than the ones presented for the SEAM assessment on a normal trajectory (see Tables 5.3 and 5.4). This is expected, as we are dealing with a trajectory less similar to the optimal one we are predicting, which introduces many irrelevant observed scales for the SEAM algorithm to deal with. The MSEs of DSG stay similar, as the actual normal trajectory did not change. The MSEs over all states are 1069.81 and 1542.60 for respectively DSG and SEAM.

Running a pair-wise Wilcoxon test against all the MSEs per state, we obtain (W = 304550, p = 4.395e-05), from which we can assume that DSG is significantly better than SEAM over all state predictions.

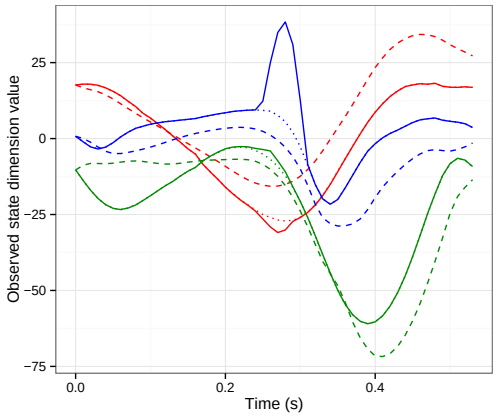


(a) Subject 1 / Trial 1 (SEAM)

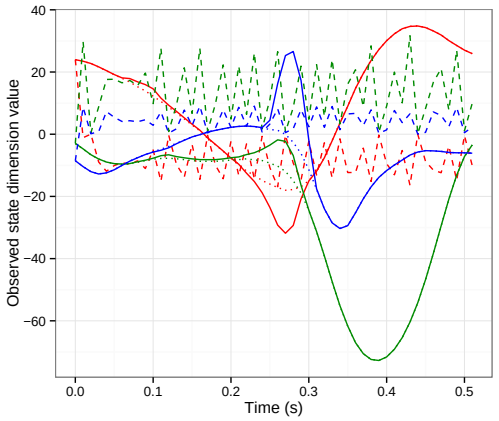
(b) Subject 1 / Trial 1 (DSG)



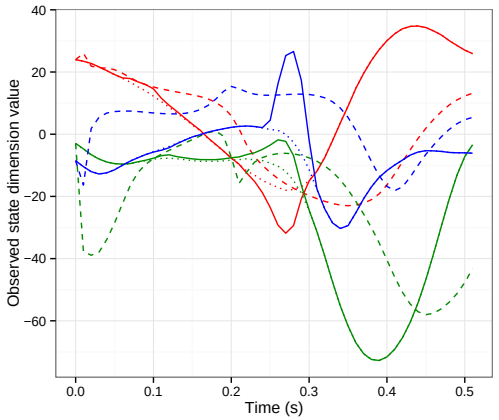
(c) Subject 2 / Trial 2 (SEAM)



(d) Subject 2 / Trial 2 (DSG)



(e) Subject 4 / Trial 2 (SEAM)



(f) Subject 4 / Trial 2 (DSG)

Figure 5.4: Predictions for SEAM (left) and DSG (right) plotted against the observed weak trajectory. The top and middle two plots respectively present the trial pairs for which SEAM and DSG performs the best. The bottom two plots shows the trials for which SEAM is the worst.

5.4.2 Discussion

First, we attempted to predict and align a normal trajectory using SEAM, while observing a person with a normal walking pattern. Afterwards, we assessed SEAM the same way, while observing a weak trajectory. We compared both methods to DSG.

Both the DSG and SEAM methods seem to be unstable, which might be an inherent property of our generative models. More specifically, as trajectories are located in a tiny, defined region of the state space for a specific motion task, a small discrepancy from the means of the latent process could have a huge impact on the behavior of the generative model. Note as well that, due to this phenomenon, the initial state of the observed trajectory (and initial scale for SEAM) has an impact on the behavior as well. If this state starts too far away from the latent normal trajectory, the variance on these latent points will be too high for the models to generate reliable state sequences.

Additionally, SEAM has the issue of estimating a model scale by attempting to solve an under-constrained problem. More specifically, learning two aspects (i.e., the scaling of and deviation from the latent normal trajectory) while observing just one data point requires splitting these two types of intertwined information, which is a hard problem to solve without an oracle that gives us feedback. This is in contrast to DSG, which only needs to worry about a proper initialization point and a model scale similar to the observed trajectory. As we both train the scale-independent normal model on, and observe, trajectories scaled by a factor of 2, this should be an additional benefit to the DSG method.

While the results for the SEAM assessment by observing a normal trajectory are inconclusive, the evaluation of SEAM while observing a weak trajectory shows that DSG performs better than SEAM. However, note that the accuracy for SEAM does not decrease that much when introducing the capability gap, which shows that the relevance of scales is considered.

Even considering all the aspects that SEAM has to worry about, it still can keep up with many trajectories compared to DSG. Once the SEAM method has been polished by for example fine-tuning the parameters $thresh_{conf}$ and $size_W$, or introducing a more method to aggregate previous scales, we could opt for SEAM instead of the less flexible DSG approach. This allows us to provide on-line information about a possible desired normal trajectory, even when the wearer of the exoskeleton changes his pace.

However, our use of generative models in an on-line support environment should be revisited, as they are too unstable to consider in practical settings.

5.5 Single-Task Support Assessment

In this experiment, we assess the learning capabilities of the single-task support model. This entails modeling the capability gaps of a specific person off-line and providing the needed support in an on-line fashion.

First, we attempt to learn a person-specific support model $\delta_p^{a^{-1}}$ off-line using $\eta^{scaled}(\cdot)$ and synthetically altered weak trajectories, after which we transform an unseen test trajectory into the desired normal trajectory. We then compare the predicted trajectory with the actual normal trajectory on which the capability gap was induced. By doing this, we evaluate the learning capabilities of our defined support model. The tasks we consider for this are 'gait' ($a = 1$), 'sit to stand' ($a = 2$) and 'stair ascent' ($a = 3$). We use all the trials available in the ADL data set and pick for each subject 2 training trajectories and 1 test trajectory according to a 3-fold cross validation scheme.

Next, we run SEAM on the same weak test trajectories, while applying our trained support models on the observed weak states. This brings the observed trajectory closer to the desired trajectory, which should make it easier for SEAM, as we are more confident in the extracted scales.

We use the same scale-dependent normal models as described in Section 5.4. Additionally, when learning or applying a model $\delta_p^{a^{-1}}$, we make sure that person p is not included in the training set of $\eta^{scaled}(\cdot)$. We focus the 'gait' task and pick the first 5 subjects of the ADL data set, which each have 3 trajectories. Again, we use a 3-fold cross validation scheme to split these into a training set and a test set.

In both experiments, we use the $k_m(\cdot)$ kernel with $k_s(\cdot) = k_{SE}(\cdot)$ and use the same configuration for SEAM as in Section 5.4. Moreover, we optimize our hyperparameters using 100 iterations of CGD on each parameter, using all points in the data set. An example of hyperparameters for a single-task support model (for subject 2, trial 3) is given below

- $\sigma \rightarrow 0.14$
- $\boldsymbol{\lambda} \rightarrow [18.59, 8.63, 10.18]$
- $\mathbf{h_l} \rightarrow [2.19, 2.76, 2.94]$
- $\boldsymbol{\theta_c} \rightarrow \begin{bmatrix} 6.51 & -5.65 & 0.25 \\ -5.65 & 5.16 & -0.46 \\ 0.25 & -0.46 & 3.09 \end{bmatrix}$

$\delta^{1^{-1}}(\cdot)$	**trial 1**	**trial 2**	**trial 3**
subject 1	8.41	1.20	0.97
subject 2	3.71	225.16	2.39
subject 3	4.64e-1	3.71e-1	3.89
subject 4	7.71	62.96	264.66
subject 5	1.77	1.50	1.75
subject 6	11.18	18.48	3.30
subject 7	39.29	4.57	8.72

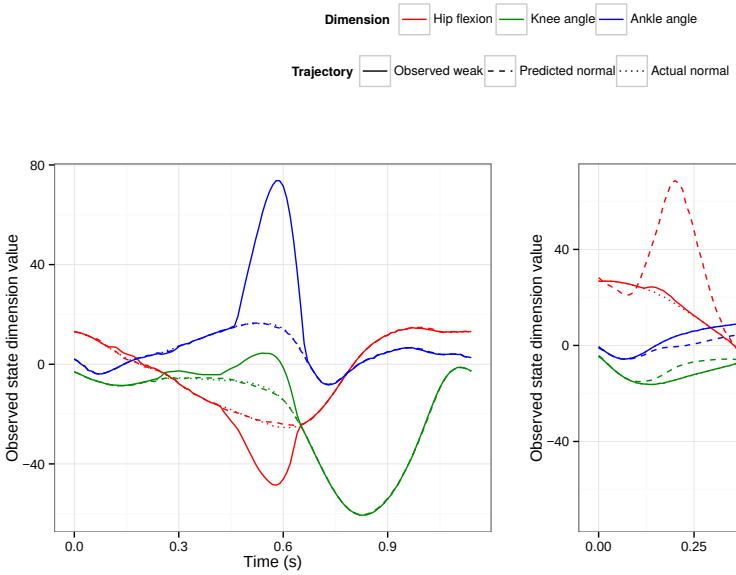
Table 5.7: MSE for each of the 21 trials in the off-line single-task support experiment. We applied $\delta^{1^{-1}}(\cdot)$ to an observed weak trajectory for activity 1 ('gait'). The highlighted entries are shown in Figure 5.5.

5.5.1 Results

Off-line Assistance

Respectively the best and worst trials for our support models are shown on the left and on the right in Figure 5.5 for each activity. At first sight, we can see that our single support model works well and that there is not a lot of room for improvement. The best predicted trajectories are almost overlapping with the actual ones. For the worst predictions, we only have three big deviations from the normal one, namely, the peak in the hip flexion in plot (b) and the two discrepancies in plot (f). Note that the capability gap induced in the trajectories of activity two is almost negligible. This makes the support function almost equal to the identity function, which is easy to learn.

The MSEs per trial are shown in Tables 5.7, 5.8 and 5.9 for respectively 'gait', 'sit to stand' and 'stair ascent'. As already mentioned, the learned support model for activity 2 is almost the identity function, which is easy to learn. We can see here that the errors for this activity are almost negligible. The gait has MSEs ranging from about 1 to about 260, which is still low. Most of these have just a small region in which poor inference is being done by the support model, but other than that, have the predicted trajectory mainly overlapping with the actual one. Activity 3 has the highest MSE values, which indicates that this support model is difficult to learn. This is possibly due to the fact that a lot of regions seem similar, e.g., the areas around 0.4 and 1.4 seconds in plot (f) of Figure 5.5. Yet, only in the area around 0.4 seconds, a capability gap is introduced, which extrapolates support to states in the area around 1.4 seconds. For activity 1, 2 and 3, we respectively have a MSE of 29.12, 0.84 and 400.71 over all observed state pairs.



(a) Task 1 / Subject 3 / Trial 2 ($\delta^{1^{-1}}(\cdot)$)

(b) Task 1 / Subject 4 / Trial 3 ($\delta^{1^{-1}}(\cdot)$)

(c) Task 2 / Subject 2 / Trial 1 ($\delta^{2^{-1}}(\cdot)$)

(d) Task 2 / Subject 7 / Trial 1 ($\delta^{2^{-1}}(\cdot)$)

(e) Task 3 / Subject 5 / Trial 1 ($\delta^{3^{-1}}(\cdot)$)

(f) Task 3 / Subject 3 / Trial 3 ($\delta^{3^{-1}}(\cdot)$)

Figure 5.5: Single-task support model (trained off-line) on observed weak trajectories. The best and worst supports provided are respectively shown on the left and right side.

$\delta^{2^{-1}}(\cdot)$	**trial 1**	**trial 2**	**trial 3**
subject 1	1.94e-2	2.33e-1	1.24e-2
subject 2	3.60e-4	1.47	3.80e-4
subject 3	2.29e-3	3.77	4.03e-3
subject 4	9.19e-3	3.73e-2	7.46e-3
subject 5	1.39e-1	1.26e-1	1.88e-3
subject 6	1.62e-3	1.76e-2	2.40e-3
subject 7	9.06	3.00	2.63

Table 5.8: MSE for each of the 21 trials in the off-line single-task support experiment. We applied $\delta^{2^{-1}}(\cdot)$ to an observed weak trajectory for activity 2 ('sit to stand'). The highlighted entries are shown in Figure 5.5.

$\delta^{3^{-1}}(\cdot)$	**trial 1**	**trial 2**	**trial 3**
subject 1	655.31	243.82	358.42
subject 2	34.46	181.57	143.59
subject 3	101.93	220.19	2065.26
subject 4	293.75	102.82	303.80
subject 5	15.29	29.72	40.36
subject 6	181.13	2695.84	45.91
subject 7	92.78	22.36	20.09

Table 5.9: MSE for each of the 21 trials in the off-line single-task support experiment. We applied $\delta^{3^{-1}}(\cdot)$ to an observed weak trajectory for activity 3 ('stair ascent'). The highlighted entries are shown in Figure 5.5.

$SEAM + \delta^{1^{-1}}(\cdot)$	**trial 1**	**trial 2**	**trial 3**
subject 1	1360.65	805.70	716.63
subject 2	1374.01	819.82	1917.37
subject 3	1162.71	1083.71	851.64
subject 4	1752.57	1576.50	1208.92
subject 5	1536.75	1409.98	1245.98

Table 5.10: MSE for each of the 15 runs of SEAM+$\delta^{1^{-1}}(\cdot)$, applied to an observed weak trajectory. Each subject has a differently trained $\eta^{scaled}(\cdot)$. The highlighted entries are shown in Figure 5.6.

Supported SEAM

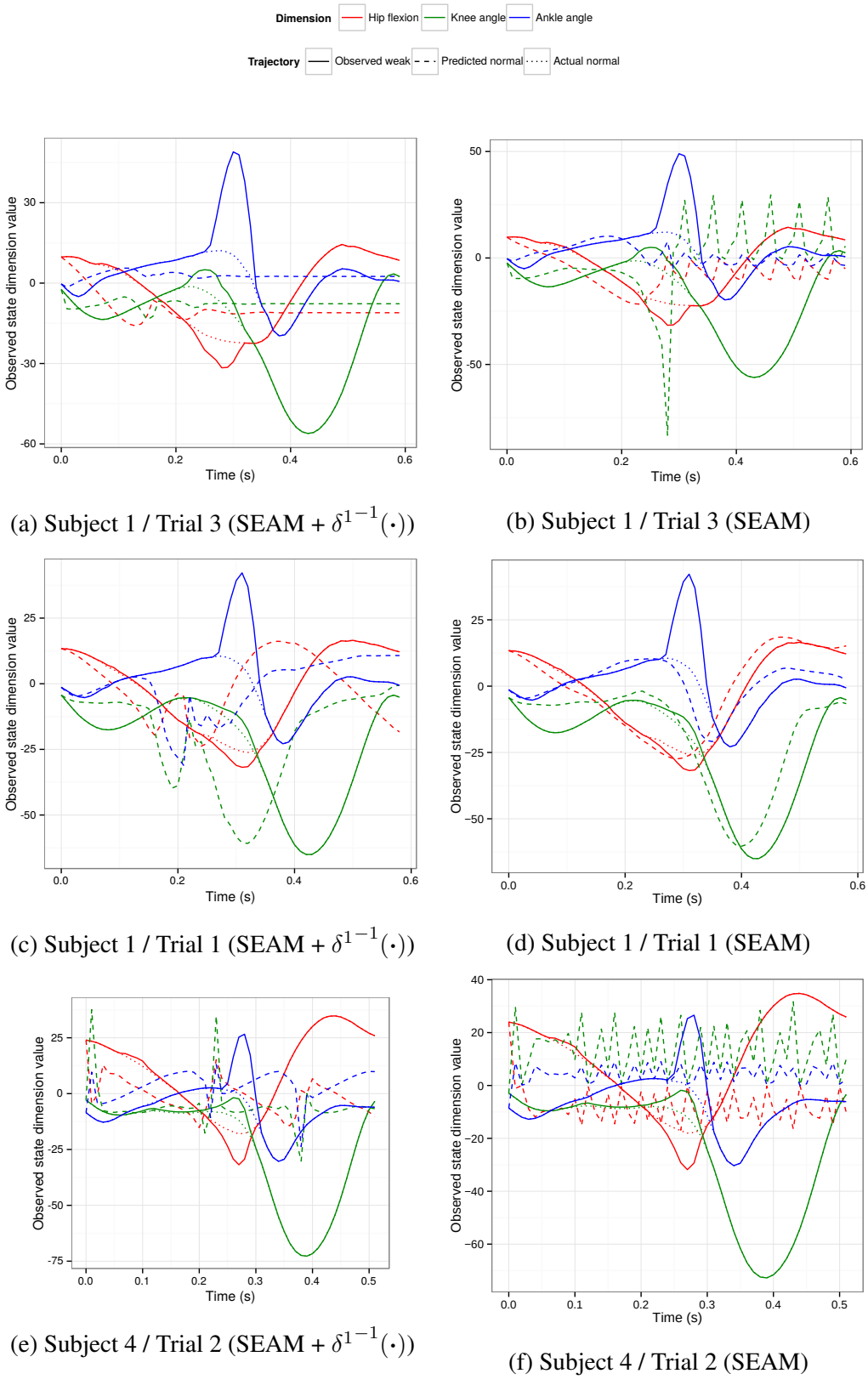
In Figure 5.6, we can see the predicted series for our support enhanced SEAM, i.e., SEAM+$\delta^{1^{-1}}(\cdot)$, and SEAM itself. First, in plot (a), we can see the best result for SEAM+$\delta^{1^{-1}}(\cdot)$, which actually converges to a constant state. In plot (b), which shows the corresponding SEAM trial, we see that SEAM manages to follow the observed trajectory, after which regular fluctuations appear. In the next two plots, (c) and (d), we can see that SEAM has accurate results, while SEAM+$\delta^{1^{-1}}(\cdot)$ manages to form a normal trajectory at the end, but with a wrong scale and phase. Plots (e) and (f) show the worst trials w.r.t. SEAM. SEAM+$\delta^{1^{-1}}(\cdot)$ does not manage to solve it.

In Table 5.10, we can find the MSEs for our support-enhanced SEAM. The results are fairly stable compared to the ones of SEAM, shown in Table 5.6. This probably indicates the tendency to move towards a constant around zero, which is shown in plot (a) of Figure 5.6. Moreover, at first sight, the DSG method used in Section 5.4 seems to outperform SEAM+$\delta^{1^{-1}}(\cdot)$. The MSE over all states are 1069.81, 1542.60 and 1242.31 for respectively DSG, SEAM and SEAM+$\delta^{1^{-1}}(\cdot)$.

By applying our pair-wise Wilcoxon tests to compare SEAM+$\delta^{1^{-1}}(\cdot)$ with DSG and SEAM, we respectively obtain $(W = 320270, p = 0.013)$ and $(W = 358970, p = 0.137)$. Thus, the results between the supported SEAM and normal SEAM are insignificant, which means we cannot tell whether SEAM with support is actually better than without. However, we can assume that the supported SEAM method performs less well than DSG over all state predictions.

5.5.2 Discussion

In an off-line environment, our single-support model seems to adjust accurately unseen weak trajectories of a specific-person to their corresponding normal trajectories, even when given a small training set of 2 trials. However, when in-



(a) Subject 1 / Trial 3 (SEAM + $\delta^{1^{-1}}(\cdot)$)

(b) Subject 1 / Trial 3 (SEAM)

(c) Subject 1 / Trial 1 (SEAM + $\delta^{1^{-1}}(\cdot)$)

(d) Subject 1 / Trial 1 (SEAM)

(e) Subject 4 / Trial 2 (SEAM + $\delta^{1^{-1}}(\cdot)$)

(f) Subject 4 / Trial 2 (SEAM)

Figure 5.6: Predictions for SEAM+$\delta^{1^{-1}}(\cdot)$ (left) and SEAM (right) plotted against the observed weak trajectory. The top and middle two plots respectively present the trial pairs for which SEAM+$\delta^{1^{-1}}(\cdot)$ and SEAM performs the best. The bottom two plots shows the trials for which SEAM is the worst.

corporated in SEAM, providing support seems to throw the method off track. Mainly, this is again due to the instability of generative models. When support is learned slightly incorrectly (e.g., at places where no support is required), the model extrapolates to unknown states, drawing it towards the mean prior and making the predicted trajectory go off track. The comparison between SEAM and SEAM+$\delta^{1^{-1}}(\cdot)$ is inconclusive. However, we can say that SEAM+$\delta^{1^{-1}}(\cdot)$ performs worse than DSG, as is the case with the normal SEAM method.

5.6 Multi-Task Support Assessment

During this experiment, we test whether incorporating multiple tasks in our support model actually improves the prediction accuracy. To do this, we learn and test our multi-task support model in an analogous way as the single-task support model and compare them both.

We use the same setting as described in Section 5.5, with the exception that we add a multi-task support model $\delta^{multi^{-1}}$. This model is trained using the same folds as each of the single-task support models, aggregated over the activities. Remember that the activities we focus on are 'gait', 'sit to stand' and 'stair ascent'. The hyperparameters for the multi-task support model associated with test trial 1 of subject 1 are

- $\sigma \rightarrow 0.0083$
- $\boldsymbol{\lambda} \rightarrow [2.73, 1.96, 1.07]$
- $\mathbf{h_l} \rightarrow [2.73, 2.95, 1.82]$
- $\boldsymbol{\theta_c} \rightarrow \begin{bmatrix} 5.37 & -4.40 & 0.22 \\ -4.40 & 4.08 & -0.23 \\ 0.22 & -0.23 & 0.39 \end{bmatrix}$
- $\mathbf{h_l^a} \rightarrow [2.08, 1.26, 5.72]$
- $\boldsymbol{\theta_c^a} \rightarrow \begin{bmatrix} 0.93 & 0.21 & -0.17 \\ 0.21 & 0.74 & -0.02 \\ -0.17 & -0.02 & 12.36 \end{bmatrix}$

We can see that the variance for activity 3 in $\boldsymbol{\theta_c^a}$ is fairly high.

5.6.1 Results

In Figure 5.7, we can see the best and worst predictions done by our multi-task support model for all activities. Except for plot (f), which has high peaks at the

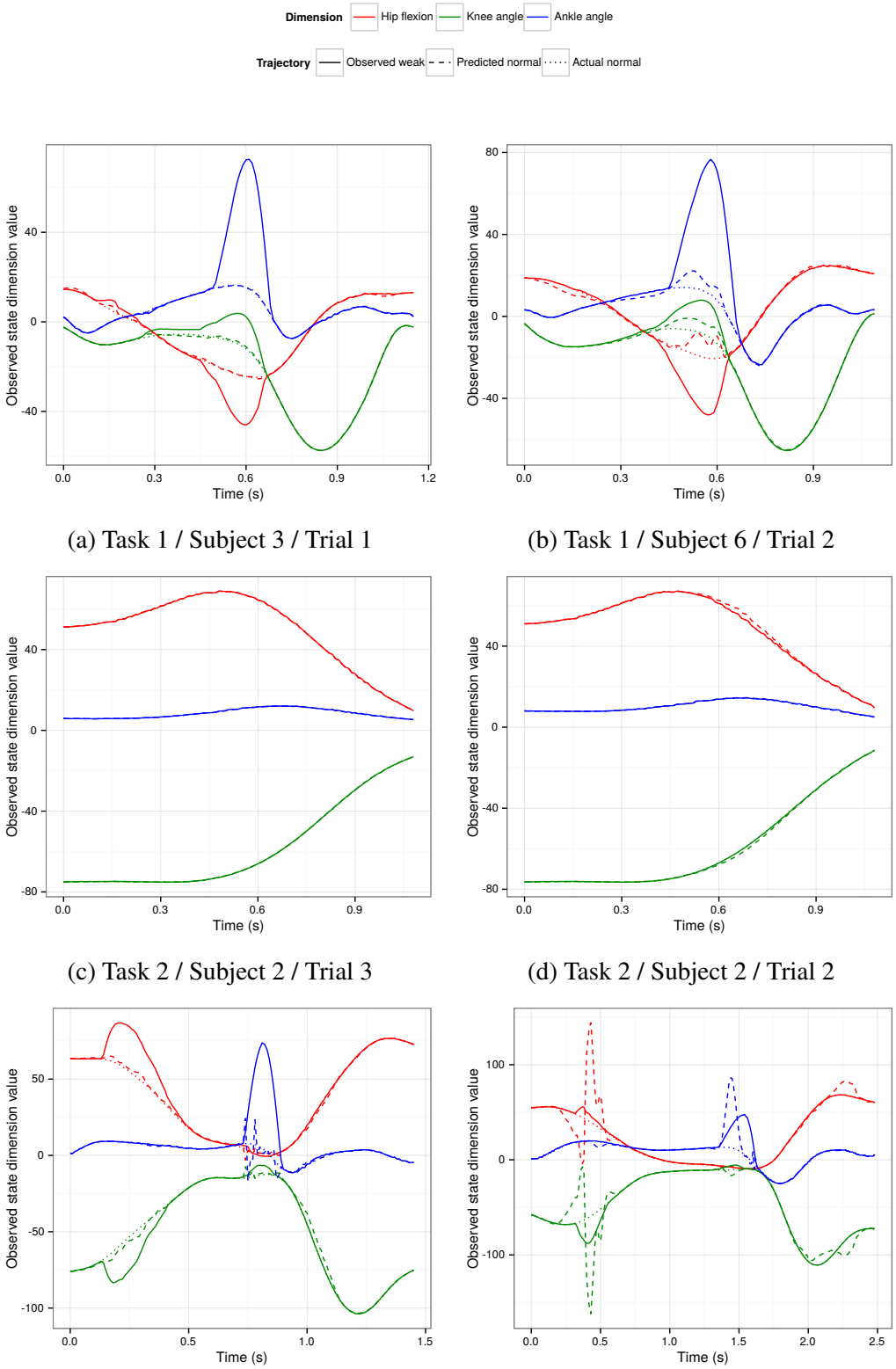
$\delta^{multi^{-1}}(\cdot)$		**trial 1**	**trial 2**	**trial 3**
activity 1	**subject 1**	76.45	1.15	3.31
	subject 2	2.77	97.31	31.14
	subject 3	1.01	7.96	5.13
	subject 4	4.20	17.16	55.81
	subject 5	28.58	59.69	21.86
	subject 6	8.58	227.88	14.59
	subject 7	58.66	53.51	8.18
activity 2	**subject 1**	0.30	2.80	3.91
	subject 2	0.07	7.50	0.01
	subject 3	0.50	6.53	4.53
	subject 4	0.11	1.19	3.09
	subject 5	4.58	0.50	0.27
	subject 6	0.23	2.60	0.07
	subject 7	0.85	0.64	2.56
activity 3	**subject 1**	43082.12	2118.34	926.71
	subject 2	131.66	369.86	504.01
	subject 3	181.30	3197.22	3203.00
	subject 4	432.29	824.59	410.98
	subject 5	21.20	44.33	8066.72
	subject 6	470.60	3253.05	43.69
	subject 7	76.12	21.57	6.77

Table 5.11: MSE for the 21 trials of each of the 3 activities in the off-line multi-task support experiment. We applied $\delta^{multi^{-1}}(\cdot)$ to an observed weak trajectory for activity 1 ('gait'), 2 ('sit to stand') and 3 ('stair ascent'). The highlighted entries are shown in Figure 5.7.

start, the support given is fairly accurate. As we can see here again, the required support for task 2 is minimal, making it easy to learn a support function.

In Table 5.11, we can see all the results aggregated per trial, for every subject and every activity. We can see that the results are similar to the ones in the single-task support experiment, given in Tables 5.7, 5.8 and 5.9, in the sense that activity 2 is easy to learn support for, activity 1 has a good accuracy and activity 3 is a bit harder to predict support for. Also, note that trial 1 of subject 1 has a high MSE, due to the peaks in plot (f) of Figure 5.7. The aggregated MSE over all state-pairs is given in Table 5.12.

Looking at the results of our pair-wise Wilcoxon test in Table 5.13, we can see that the results shown in Table 5.12 are significant, meaning that we can assume that the single-task support model works better than the multi-task one in this



(a) Task 1 / Subject 3 / Trial 1

(b) Task 1 / Subject 6 / Trial 2

(c) Task 2 / Subject 2 / Trial 3

(d) Task 2 / Subject 2 / Trial 2

(e) Task 3 / Subject 7 / Trial 3

(f) Task 3 / Subject 1 / Trial 1

Figure 5.7: Multi-task support model $\delta^{multi^{-1}}$ (trained off-line) applied on observed weak trajectories. The best and worst supports for each of the trajectories are respectively shown on the left and right side.

a	$\delta^{a^{-1}}$	$\delta^{multi^{-1}}$
1	30.96	36.83
2	0.93	2.03
3	382.70	4269.98

Table 5.12: MSE over all state-pairs for both the single-task support models $\delta^{a^{-1}}$ and the multi-task support $\delta^{multi^{-1}}$, for every activity a.

(W, p)	$\delta^{multi^{-1}}$
$\delta^{1^{-1}}$	(2194000, 2.2e-16)
$\delta^{2^{-1}}$	(1482800, 2.2e-16)
$\delta^{3^{-1}}$	(5267900, 2.2e-16)

Table 5.13: The W- and p-values for the Wilcoxon tests between $\delta^{multi^{-1}}$ and $\delta^{a^{-1}}$ for every activity a.

setting.

5.6.2 Discussion

We extended our single-task to a multi-task support process, which manages multiple tasks. This model considers correlations amongst capability gaps over multiple activities for a single person. As these different task-specific gaps are exhibited by the same physical motor dysfunction, it is expected that these correlations can induce cross-task information transfer in our model. Therefore, the management of multiple tasks in a single model prevents redundancy, in contrast to our many single-task support models. Additional to that, it is easy to maintain one support model, as it is easily extensible with a new task, given that a normal model is available for that new task. Moreover, the multi-task support model has the ability to extrapolate/interpolate to other unseen tasks, as the weakness that has to be supported is mainly dependent on the pose in the trajectory and will thus approximately occur at the same states over multiple tasks.

Our results show that single-task support model is significantly better than the multi-task support model, despite the cross-task transfer. This is probably due to the lack of training data and the fact that there is not a lot to improve on w.r.t. the learning capabilities of the single-task model. Another possibility is that the multi-task model is too general. For example, the model uses the same hyperparameters for each task, which requires every activity to operate on the same length scales. Comparing activity 1 to activity 2, we can see that the former has a smaller length scale than the latter (see Figure 5.7), which could reduce

the prediction capabilities of the multi-task support model. Choosing a better covariance kernel for the domain of application could yield better results.

Even though our results show that our single-task support model is significantly better than the multi-task one, the difference in performance is small enough to possibly opt for the latter one, given the benefits mentioned earlier.

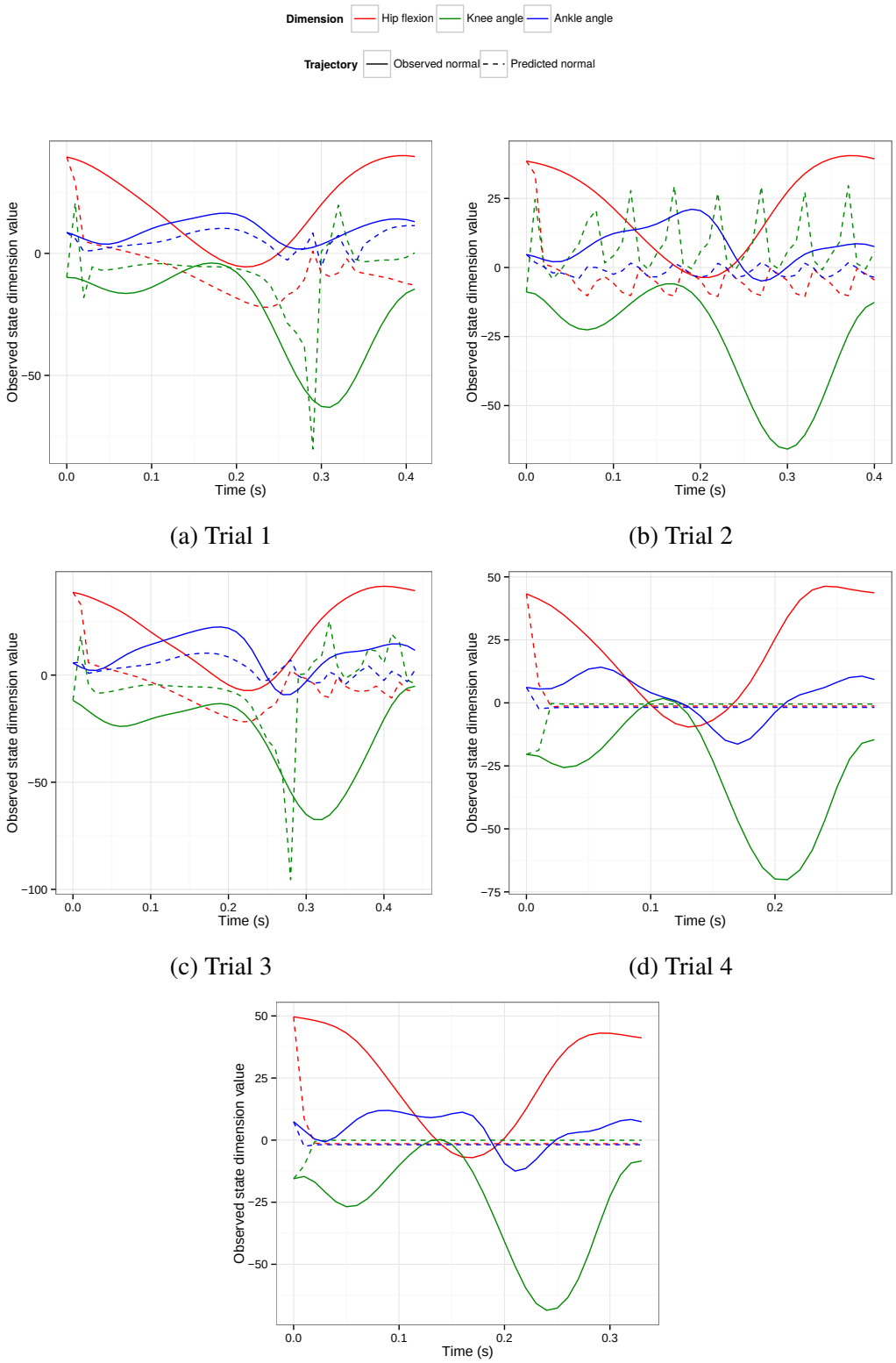
5.7 Children's Gait Support

Finally, we apply our support framework in a real-world setting. This examination is rather a proof of concept than an experiment, as we are using real-world weak trajectories and have nothing to compare their predicted normal trajectories against. Moreover, presenting the application of our support framework to a practical setting might give us more insight on what we could improve on. We attempt to provide support to children, in order to adjust their irregular walking pattern to a regular gait cycle. As mentioned before, this is not particularly representative for paraplegics. However, due to the instability of SEAM, we should push our boundaries and first test the method in a simple real-world settings.

We learn our single-task support model $\delta^{-1}(\cdot)$ on the children's gait data set. We test our framework on 3 children, each having 5 trials. Even though we want to test our framework in a practical setting, where manipulation of observed trajectories is impossible, we still adjust the used gaits to be twice the speed, as explained in Section 3.3.2. Assuming these trajectories have the same scaling over time as the ones in the ADL data set, this is necessary to make sure our model interpolates to unknown scales, rather than extrapolates, as we train our scale-dependent normal models to once, twice and thrice the actual speed. However, such manipulation can be done indirectly in a practical setting by requiring the humans to walk at different paces and train our model on those trajectories, such that interpolation between these paces can be done.

The first 3 trials are executed without support (i.e., only SEAM running over them). The 4^{th} trial uses a support model trained on the observed-predicted state pairs found during the first 3 trials. Meanwhile, we add the newly found mappings in the 4^{th} trial to the support model and run a final trial using the updated model. As a generative normal model, we use the best performing one during the SEAM assessment on weak trajectories, namely, $\eta^{scaled}(\cdot)$ for subject 1. As the results for every subject are similar, we focus specifically on subject 2, as the behavior of the support framework is the clearest for these trials.

Looking at Figure 5.8, we can see the five trials performed by subject 2, along with a predicted normal trajectory. As expected, the SEAM method is unstable in the first three trials. In trial 1 and 3, we can see that it can estimate the model scale pretty well until 0.25 seconds, after which it loses grasp of the underlying desired



(a) Trial 1

(b) Trial 2

(c) Trial 3

(d) Trial 4

(e) Trial 5

Figure 5.8: Children's gait trials for subject 2. The SEAM method is unsupported for the first three trials. The fourth trial uses the support model learned over the previous three trajectories. The final trial uses a support model containing information gathered over the other four trials.

normal trajectory. The last two trials, for which support of the previous trials is incorporated, fails completely due to the poorly learned support.

6

CONCLUSION

In order to build an exoskeletal-assisted framework, we began our journey by studying Gaussian processes, which lie at the basis of our modeling and prediction schemes. A Gaussian process is a Bayesian inference technique that characterizes a distribution over functions. Performing regression using these processes provides us with a flexible non-linear modeling approach for high-dimensional spatio-temporal motion data. As we are dealing with a dynamic system of pose transitions, we introduced multi-output dynamical GPs, which allow us to model these transitions and generate artificial state sequences.

Next, it was necessary to formalize the models, instances of them and the relationships amongst them. We introduced normal and weak processes, in order to characterize respectively trajectories which meet the task's requirements and those which do not. Afterwards, we could define a reduction model that captured the capability gaps induced upon the human-specific weak processes. In terms of this, we formalized a support process, which models the adjustment of these gaps. Additionally, we introduced scale-dependency in our processes, such that trajectories performed at various paces, could be modeled.

Then, we described our methods for learning a support model in an on-line environment. We presented a novel method, namely the Scale Extraction and Adaptation Method, or SEAM, which attempts to extract the desired scale required for normal state generation from observed weak poses. This gives us aligned trajectories, from which a mapping from weak to normal trajectory could be established and henceforth gives rise to a support model. We extended our support process formalism to multiple tasks, in order to manage them conjointly. This construction

is driven by the idea that the capability gaps over multiple motion tasks are due to the same motor dysfunctions, which makes us believe these are highly correlated.

Finally, we evaluated the aspects that build up our framework. SEAM seems to be an unstable method, which is mainly inherent to generative models. However, in contrast to scale-independent approaches, it is able to change pacing at any moment, which makes it more flexible in a practical setting. Our support models grasp the capability gaps well and can provide the necessary assistance. An additional benefit to the multi-task approach, is that it allows for extrapolation to other tasks, which, even though single-task support is significantly better, provides a cleaner approach to model capability gaps and prevents redundancy of information. We concluded with a practical experiment, in which children's gaits were considered. These are assumed to be irregular, and therefore formed a simple playground for our methods to be tested on. In the end, our framework did not manage to provide proper support in this setting. However, we gained valuable insights on what requirements should be imposed on exoskeleton-driven support frameworks, what is applicable in these type of settings and what could be improved upon.

6.1 Future Work

Various extensions upon this work can be done. During our experiments, we assume we know what activity we are observing. Yet in a practical setting, the exoskeleton has to infer which one this is. Hidden Markov Models can be incorporated in our framework, in order to recognize the task under observation, after which support can be induced accordingly.

As our framework is general enough to be disconnected from the field of human motion, we can apply it to other areas of research. In general terms, our multi-task support model encourages the idea of capturing correlations among deviations from certain optima and extrapolate that knowledge to other instances of that. Therefore, this model could be applicable to any domain which has unsolved problems closely related to a group of solved ones. For example, in the domain of reinforcement learning, when we want agents to learn a specific value function similar to a group of previously learned related value functions, we could guide an agent towards the optimal path, given that we have support models available for this group of related value functions.

Finally, this method gave valuable insights on the requirements for exoskeleton-driven support. These findings could therefore prove useful to similar research, e.g., performed by MIRAD. As we adopted another perspective than a biomedical one, our findings could be inspirational when they start incorporating on-line learning capabilities into their exoskeleton.

BIBLIOGRAPHY

Afschrift, M., De Groote, F., De Schutter, J., and Jonkers, I. (2014). The Effect of Muscle Weakness on the Capability Gap During Gross Motor Function: A Simulation Study Supporting Design Criteria for Exoskeletons of the Lower Limb. *BioMedical Engineering OnLine*, 13.

Álvarez, M. and Lawrence, N. D. (2009). Sparse Convolved Gaussian Processes for Multi-Output Regression. In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems*, volume 21, pages 57–64. Curran Associates, Inc.

Álvarez, M. A., Luengo, D., Titsias, M. K., and Lawrence, N. D. (2010). Efficient Multi-Output Gaussian Processes through Variational Inducing Kernels. In Whye Teh, Y. and Titterington, M., editors, *Proceedings of The Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 25–32, Chia Laguna Resort, Sardinia, Italy.

Banerjee, A., Dunson, D. B., and Tokdar, S. T. (2013). Efficient Gaussian Process Regression for Large Datasets. *Biometrika*, 100:75–89.

Banerjee, S., Gelfand, A. E., Finley, A. O., and Sang, H. (2008). Gaussian Predictive Process Models for Large Spatial Data Sets. *Journal of the Royal Statistical Society*, 70:825–848.

Bonilla, E. V., Chai, K. M. A., and Williams, C. K. I. (2008). Multi-Task Gaussian Process Prediction. *Advances in Neural Information Processing Systems*, 20:153–160.

Boyle, P. and Frean, M. (2005). Dependent Gaussian Processes. *Advances in Neural Information Processing Systems*, 17:217–224.

Candela, J. Q. n., Rasmussen, C. E., and Williams, C. K. I. (2007). Approximation Methods for Gaussian Process Regression. In Bottou, L., Chapelle, O., DeCoste, D., and Weston, J., editors, *Large-Scale Kernel Machines*, pages 203–224. The MIT Press, Cambridge, MA, USA.

Cunningham, J. P., Ghahramani, Z., and Rasmussen, C. E. (2012). Gaussian Processes for Time-Marked Time-Series data. *Journal of Machine Learning Research*, 22:255–263.

Fasshauer, G. E. (2011). Positive Definite Kernels: Past, Present and Future. *Dolomites Research Notes on Approximation*, 4:21–63.

Grinstead, C. M. and Snell, J. L. (1988). *Introduction to Probability*, chapter 11, pages 405–470. American Mathematical Society, 2 edition.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, chapter 8, pages 267–270. Springer Series in Statistics. Springer, New York, NY, USA, 2 edition.

Lancaster, H. (2006). Chi-Square Distribution. In Kotz, S., editor, *Encyclopedia of Statistical Sciences*. Wiley, New York, NY, USA, 2 edition.

Lawrence, N. (2005). Probabilistic Non-Linear Principal Component Analysis with Gaussian Process Latent Variable Models. *Journal of Machine Learning Research*, 6:1783–1816.

Lawrence, N. D. (2004). Gaussian Process Latent Variable Models for Visualisation of High Dimensional Data. In Thrun, S., Saul, L. K., and Schölkopf, B., editors, *Proceedings of Advances in Neural Information Processing Systems 16 (NIPS 2003)*, pages 329–336, Cambridge, MA, USA. The MIT Press.

Markov, A. A. (1906). Extension of the Law of Large Numbers to Dependent Quantities. *Izvestiia Fiz.-Matem. Obsch. Kazan Univ.*, 2:135–156.

Martín-F'elez, R., Mollineda, R. A., and Sánchez, J. S. (2011). Human Recognition Based on Gait Poses. *Lecture Notes in Computer Science*, 6669:347–354.

Medical Expo (2014). Robotic Exoskeleton for Assisted Walking. [Online; accessed 15-August-2015].

Melkumyan, A. and Ramos, F. (2011). Multi-Kernel Gaussian Processes. In Walsh, T., editor, *Proceedings of the 21th International Joint Conference on Artificial Intelligence (IJCAI 2011)*, pages 1408–1413. AAAI Press.

MIRAD (2014). MIRAD: An Integrated Methodology to Bring Intelligent Robotic Assistive Devices to the User. http://www.mirad-sbo.be/ [Online; accessed 23-August-2015].

Mukherjee, B. N. and Maiti, S. S. (1988). On Some Properties of Positive Definite Toeplitz Matrices and their Possible Applications. *Linear Algebra and its Applications*, 102:211–240.

Osborne, M. A., Roberts, S. J., Rogers, A., and Ramchurn, S. D. (2008). Towards Real-Time Information Processing of Sensor Network Data Using Computationally Efficient Multi-Output Gaussian Processes. In *International Conference on Information Processing in Sensor Networks (IPSN 2008)*, pages 109–120, 22–24 April, St. Louis, MO, USA.

Pinheiro, J. C. and Bates, D. M. (1996). Underconstrained Parameterizations for Variance-Covariance Matrices. *Statistics and Computing*, 6:289–296.

Rabiner, L. R. (1989). A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. In *Proceedings of the IEEE*, volume 77, pages 257–286. IEEE.

Rasmussen, C. E. (2000). The Infinite Gaussian Mixture Model. In Solla, S., T.K., L., and Müller, K.-R., editors, *Proceedings of Advances in Neural Information Processing Systems (NIPS 1999*, volume 12, pages 554–560. The MIT Press, Cambridge, MA, USA.

Rasmussen, C. E. (2004). Gaussian Processes in Machine Learning. In *Advanced Lectures on Machine Learning*, pages 63–71. Springer.

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. The MIT Press, Cambridge, MA, USA.

Roberts, S., Osborne, M., Ebden, M., Reece, S., Gibson, N., and Aigrain, S. (2012). Gaussian Processes for Time-Series Modelling. *Philosophical Transactions A*, 371.

Rosenblatt, M. (2000). *Gaussian and Non-Gaussian Linear Time Series and Random Fields*. Springer-Verlag, New York, NY, USA.

Schnabel, R. B. and Eskow, E. (1990). A New Modified Cholesky Factorization. *SIAM Journal on Scientific and Statistical Computing*, 11:1136–1158.

Seeger, M. (2008). Low Rank Updates for the Cholesky Decomposition. Technical report, University of California, Berkeley, CA, USA.

Shewchuk, J. R. (1994). An Introduction to the Conjugate Gradient Method Without the Agonizing Pain. Technical report, Carnegie Mellon University, Pittsburgh, PA, USA.

Smith, L. I. (2002). A Tutorial on Principal Components Analysis. Technical report, Cornell University, Ithaca, NY, USA.

Tolani, D., Goswami, A., and Badler, N. I. (2000). Real-Time Inverse Kinematics Techniques for Anthropomorphic Limbs. *Graphical Models*, 62:353–388.

Wang, J. M., Fleet, D. J., and Hertzmann, A. (2008). Gaussian Process Dynamical Models for Human Motion. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30:283–298.

Williams, C. and Barber, D. (1998). Bayesian Classification with Gaussian Processes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20:1342–1351.

Williams, C. K. I. and Rasmussen, C. E. (1996). Gaussian Processes for Regression. In Touretzky, D. S., Mozer, M. C., and Hasselmo, M. E., editors, *Proceedings of Advances in Neural Information Processing Systems (NIPS 1995)*, volume 8, pages 514–520. The MIT Press, Cambridge, MA, USA.

Zatsiorsky, V. M. and Prilutsky, B. I. (2012). *Biomechanics of Skeletal Muscles*. Human Kinetics.