

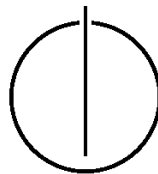
FAKULTÄT FÜR INFORMATIK

DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

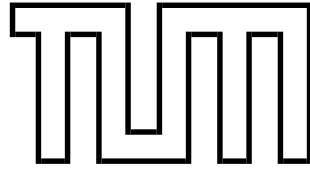
Master's Thesis in Robotics, Cognition, Intelligence

# **Online Activity Recognition from skeletal features through Kernel Methods**

Evgeni Pavlidis







FAKULTÄT FÜR INFORMATIK

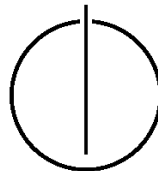
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Master's Thesis in Robotics, Cognition, Intelligence

Online Activity Recognition from skeletal features  
through Kernel Methods

Echtzeit Aktivitätserkennung durch Skelettmerkmale  
mittels Kernelmethoden

Author: Evgeni Pavlidis  
Supervisor: Prof. Dr. Daniel Cremers  
Advisor: Dr. Rudolph Triebel  
Date: October 10, 2014





---

## **Abstract**

English abstract



---

## **Zusammenfassung**

German Abstract





Ich versichere, dass ich diese Masterarbeit selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den October 10, 2014

Evgeni Pavlidis



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Motivation . . . . .	1
1.1.1	The SPENCER project . . . . .	1
1.2	Problem statement . . . . .	1
1.2.1	Online learning of human activities . . . . .	2
1.2.2	Evaluate Gaussian processes against different ml algorithms for activity recognition . . . . .	2
1.2.3	Scope . . . . .	2
1.3	Prerequisites and notation . . . . .	2
1.3.1	Mathematical notation . . . . .	3
1.4	Outline . . . . .	3
<b>2</b>	<b>Background</b>	<b>5</b>
2.1	Machine Learning . . . . .	5
2.1.1	Supervised learning . . . . .	5
2.1.2	Unsupervised learning . . . . .	6
2.1.3	Generative models . . . . .	6
2.1.4	Discriminative models . . . . .	7
2.1.5	Online learning . . . . .	7
2.1.6	Active learning . . . . .	7
2.2	Kernel methods . . . . .	7
2.2.1	A space defined by sample similarity . . . . .	7
2.2.2	The Kernel trick . . . . .	8
2.2.3	The Radial Basis Function . . . . .	8
2.2.4	Support Vector Machines . . . . .	8
2.3	Gaussian Processes . . . . .	9
2.3.1	The Gaussian distribution . . . . .	10
2.3.2	Kernels . . . . .	11
2.3.3	Regression . . . . .	11
2.3.4	Learning . . . . .	11
2.3.5	Classification . . . . .	12
2.3.6	Advantages . . . . .	12
2.3.7	Disadvantages . . . . .	12
2.3.8	Sparse Methods . . . . .	13
2.4	Gaussian Process - Latent Variable Model . . . . .	13

2.4.1	Dual Probabalistic PCA . . . . .	13
2.4.2	Back-constraints GP-LVM . . . . .	14
2.4.3	Discriminative GP-LVM . . . . .	14
2.4.4	Other variants . . . . .	15
2.4.5	Advantages . . . . .	15
2.4.6	Disadvantages . . . . .	15
2.5	Sequence similarity measures . . . . .	16
2.5.1	Dynamic Time Warping . . . . .	16
2.5.2	Longest Common Subsequence . . . . .	16
<b>3</b>	<b>Approach: KMeans clustering approach</b>	<b>19</b>
3.1	Cornell Daily Living Activities dataset . . . . .	20
3.2	Robot Operating System (ROS) . . . . .	20
3.3	Implementation . . . . .	21
3.4	Integration into ROS . . . . .	21
3.5	Shortcomings . . . . .	21
3.6	Evaluation . . . . .	22
<b>4</b>	<b>Approach: Discriminative Sequence Back-Constrained GP-LVM</b>	<b>23</b>
4.1	Discriminative Sequence Back-Constrained GP-LVM . . . . .	23
4.1.1	Sequence back-constraints . . . . .	24
4.1.2	Discriminative GP-LVM . . . . .	25
4.1.3	Advantages . . . . .	25
4.1.4	Shortcomings . . . . .	26
4.1.5	Extensions: . . . . .	26
4.2	Feature extraction . . . . .	26
4.3	Implementation . . . . .	28
4.4	Dynamic time warping with mahalanobis distance . . . . .	28
4.4.1	Implementation . . . . .	29
4.5	Evaluation . . . . .	29
<b>5</b>	<b>Approach: GP-Latent Motion Flow</b>	<b>31</b>
5.1	Gaussian Process Regression Flow . . . . .	31
5.2	GP-Latent Motion Flow . . . . .	32
5.3	Requirement: Smooth latent space . . . . .	33
5.4	Learning the flow field . . . . .	33
5.4.1	Effects of the hyperparameters . . . . .	33
5.5	Interpretation . . . . .	34
5.6	Recognition . . . . .	34
5.7	Advantages . . . . .	34
5.7.1	Recognition . . . . .	34
5.7.2	Prediction . . . . .	34
5.7.3	Multiple Hypothesis Prediction . . . . .	35

5.7.4	Online learning . . . . .	35
5.7.5	Active learning . . . . .	35
5.7.6	Natural interpretation . . . . .	35
5.7.7	Novelty detection (anomaly detection) . . . . .	35
5.7.8	In comparison to the GPDM it can model cyclic activities . . . . .	35
5.8	Implementation . . . . .	35
5.9	Problems . . . . .	35
5.9.1	Dimensionality reduction . . . . .	35
5.9.2	Stable class mean flow field . . . . .	35
5.10	Evaluation . . . . .	36
<b>6</b>	<b>Conclusions and Outlook</b>	<b>39</b>
6.1	Summary . . . . .	39
6.2	Lessons learned . . . . .	39
6.2.1	Dimensionality reduction for all activities is very difficult (also with extra constraints) . . . . .	39
6.2.2	Dynamics is a good measure for classification of human activities . . . . .	39
6.3	Contributions . . . . .	39
6.3.1	Implementation of . . . . .	39
6.3.2	Advantages and Disadvantages of dimensionality reduction with GP-LVM for human motion in the context of activity recognition . . . . .	39
6.3.3	Implementation of the Discriminative GP-LVM with python . . . . .	39
6.3.4	Implementation of the Sequence Back-constraints . . . . .	39
6.3.5	Improvement of the DTW measure with the mahalanobis distance . . . . .	39
6.3.6	A novel approach for activity recognition . . . . .	39
6.3.7	Introduction of an energy minimization approach for online recognition of complex activities . . . . .	40
6.4	Outlook . . . . .	40
6.4.1	Implementation of the GP-LMF using spatio-temporal GP-LVM . . . . .	40
6.4.2	Semi-supervised activity learning by automatic segmentation of activities . . . . .	40
	<b>Bibliography</b>	<b>43</b>



# 1 Introduction

---

## 1.1 Motivation

Activity recognition is a big research field in machine learning and robotics. Being able to infer what human actors are doing helps in many practical robotic tasks. An example is doing short-time prediction for collision-avoidance. Moreover in social robotics it is crucial to know what humans are doing when reasoning about the current state of the robot's environment.

To do activity recognition the human pose has to be inferred for each frame. With the advent and further development of RGBD (color and depth) sensors it is now possible to perform skeletal tracking of persons. This allows us to decouple pose estimation and activity learning, which make the problem a bit easier.

For real robotic tasks it is very important that the activity recognition is online i.e. runs in real-time.

Advantages of skeletal features are that no person sensitive information has to be processed by the learning algorithm. In the context of the SPENCER project this is an important prerequisite. Another advantage is also that the skeletal features are very informative for activity recognition. Also beginning to concept an algorithm that builds on top of robust pose estimation reduces the complexity, as we can fully ignore the pose estimation problem.

### 1.1.1 The SPENCER project

With the modern technologies (ms Kinect SDK, primesense ...) it is possible to decouple skeleton tracking and learning

## 1.2 Problem statement

---

cite:software packages and tools used

cite:datasets (mocap, daily activities, ms activities)

Check bibliography style and data!!!

define simple variables mathematically eg. feature sequence etc.

make a distinction between action and activity

make a distinction between online recognition and online learning !!! maybe change online to real-time

The biggest challenge in activity recognition is classification of time series data. In contrast to the simpler sample model, here we have to classify sequences. Therefore appropriate models have to be implemented that take the dynamics into account.

On top of that the problem is ill-posed.

Also for online activity recognition we have to classify incomplete data consisting of subsequences. This makes the problem more challenging and reduces the pool of methods that can be used for the classification.

We identify three requirements for a practical activity recognition system:

1. Online recognition

The algorithm has to be able to do real-time classification as the sequence is progressing.

2. Classification of incomplete sequences

This follows directly from the first requirement. There should be no assumptions about the activities regarding completeness, length or periodicity.

3. Novelty detection

The algorithm should be able to recognize unobserved activities.

elaborate  
on this

Optimally it should be possible to also recognize when an unknown activity started and finished. This way automatic segmentation will be possible and will considerably reduce the supervised learning time. In combination with active learning this will greatly reduce training time for practical applications.

Nice to have... online learning??

### 1.2.1 Online learning of human activities

Use Gaussian processes to learn new activities in real time

### 1.2.2 Evaluate Gaussian processes against different ml algorithms for activity recognition

Evaluate the performance of GPs in relation to the other solutions

### 1.2.3 Scope

## 1.3 Prerequisites and notation

We assume a basic understanding in *Linear Algebra* and *Probability theory*. Although a high-level overview on machine learning is given in chapter 2 deeper knowledge in this field will help understand the *Related Works* chapter better.



### 1.3.1 Mathematical notation

- Matrices uppercase
- Vectors lowercase bold
- Constants lowercase
- Parameters lowercase greek letters

## 1.4 Outline

**Introduction** This chapter introduced the topic of this work. The motivation and the scope is explained.

**Background** The second chapter summarizes some basic concepts and models that are prerequisites for our approach. It begins with an overview of machine learning and introduces the multivariate Gaussian distribution. Then an emphasis is led on Gaussian Process Regression and Gaussian Process - Latent Variable Models, which is an unsupervised learning method for dimensionality reduction. Last the Dynamic Time Warping algorithm, which is used for sequence alignment, is explained.

**Related Work** The third chapter gives an overview of methods used in similar approaches and then analyses strength and weaknesses of these methods in regards to online activity recognition.

**Approach** The fourth chapter presents two approaches to online activity recognition and their implementations. The first one is an implementation of "Discriminative Sequence Back-constrained {GP}-{LVM} for {MOCAP} based Action Recognition" [1]. The second one is a novel approach which learns a dense motion flow field in latent space through Gaussian Process Regression.

**Evaluation** In the fifth chapter the two approaches are being evaluated and discussed.

**Results and Outlook** The last chapter summarizes the results of the two approaches and gives a brief outlook of future improvements.



## 2 Background

This chapter introduces some basic concepts needed to understand the proposed approaches. First a high-level overview is given on machine learning and its terminology. Then the Kernel function is explained along with the *Support Vector Machine* - a kernelized learning method. Following is an explanation of *Gaussian Processes*, their different interpretations and properties. After that the *Gaussian Process - Latent Variable Model* is being introduced along with some extensions for learning a backward mapping and optimizing it for discrimination in the case of multiple classes. Last two *Sequence similarities measures* are presented which are used in our implementations.

### 2.1 Machine Learning

Machine Learning is a discipline where one makes inference on real world data. Data consists of different samples

#### 2.1.1 Supervised learning

Supervised learning is the task of classification or regression when the data is labeled i.e. we have the ground truth of every sample. The algorithm then takes the labeled samples (and maybe some confidence values) and infers the model parameters (or hyperparameters) accordingly.

There are two distinct cases in supervised learning:

##### 1. Classification

Classification is the task of learning which category a sample belongs to. A prominent example is Spam filtering. By taking a large number of emails which are labeled either as spam or as ham (regular email), the algorithm deduces a model which can classify unknown samples into these two categories.

##### 2. Regression

Regression is a terminus in machine learning and means function approximation. Here the domain of the sample's label is continuous. An example would be ...

In most cases we search for a good model that explains the data we have. Parametric models, for example, have a pre-defined model which is parametrized.

When searching for an appropriate model it is also important that we try to capture the underlying relationship without compromising the generalization property, which is the ability of the model to correctly predict unseen samples. The case that an algorithm learns the relationship of the data that is used to train the model (training data) but poorly predicts new samples is called overfitting.

Very often the parameter search is done by maximizing the probability of the data given the model parameters.

$$\arg \max_{\theta} p(\mathbf{X}|\theta) = \arg \max_{\theta} \frac{p(\theta|\mathbf{X})p(\mathbf{X})}{p(\theta)}$$

where  $\theta$  are the model parameters and  $X$  is the data.

### 2.1.2 Unsupervised learning

In contrast to supervised learning in unsupervised learning we have no labeled data i.e. there is no supervisor giving each sample a category (classification) or a value (regression). In this case we can only derive properties of the generation process. Therefore we try to detect patterns in the unlabeled data. These pattern may be clusters of similarity or a lower dimensional generative manifold from which the samples are generated. The last one is called *Dimensionality Reduction* which will be also a subject in this work. [2]

#### K-means algorithm

An example of an unsupervised learning method for finding a given number of clusters  $k$  in given data is the *k-means* method. The idea is that we first determine the number of clusters and choose  $k$  points randomly in the space, which represent a guess of the cluster means (center of mass). After that we try to move these points, such that they align with the real data's  $k$  centers of mass. This is done by iterating between two steps:

1. Assign each point  $x$  to the closest centroid (cluster mean)
2. Find new centroids by computing the mean of all assigned points for each cluster  $k$

Doing so it is guaranteed that the algorithm will converge, although it could be in a local minimum.

[2]

### 2.1.3 Generative models

Generative methods model the underlying process which generates the data. In Bayesian terms we model the likelihood and the . Thus more data is needed to find an appro-

priate model. On the other side the model is very flexible and many attributes have a natural interpretation. An example of this is

generative  
model ex-  
ample

#### 2.1.4 Discriminative models

A discriminative model is only concerned with modeling the actual posterior. This way fewer samples are needed to find an appropriate model. On the other hand by not taking the prior into account the model's ability to generalize unseen data is worse. For this reason discriminative methods are more susceptible to overfitting.

#### 2.1.5 Online learning

Algorithms which can be gradually optimized towards a good solution using streaming batches of samples are considered to do online learning. In contrast to online learning online recognition means that the algorithm works in real-time and fast recognition is possible.

#### 2.1.6 Active learning

Very often the bottleneck of powerful supervised learning techniques is that they rely on a large number of correctly labeled data. Since labeling has to be performed by a human it is very difficult and costly to label large amount of data. By identifying more important samples by their information ability of selecting a good model, it is possible to achieve good results with fewer samples. Letting the algorithm select such samples and query only their labels from a human, who is now actively participating in the learning loop, is called active learning.

Active learning is in practice a convenient way to acquire new informative samples without letting someone go over a huge amount of data to label.

## 2.2 Kernel methods

Many machine learning algorithms work not with the features directly but instead use only the dot product between features. The dot product between two vectors can be seen as a measure of similarity.

### 2.2.1 A space defined by sample similarity

Suppose we have  $n$  sample points  $x_i$  of dimensionality  $d$ :  $x_i \in \mathcal{R}^d$ . When extracting features we try to capture the most characteristic properties of the data for each sample. Let us say that we want to extract  $m$  features. Then we have a vector  $z_i \in \mathcal{R}^m$  which represents each sample. This means that learning is done in a feature space of dimensionality  $m$ . Another space, where we can reason about the data is a similarity space.

Suppose we have a function  $k(\mathbf{x}, \mathbf{y})$  which measures the similarity between point  $\mathbf{x}$  and point  $\mathbf{y}$ , then we can define a vector  $\mathbf{s}$  of similarities for a new point

This similarity measure is also called a *kernel function*. We can also define some properties of the kernel function resulting of the informal introduction as a similarity measure.

### 2.2.2 The Kernel trick

A kernel defines a similarity measure between two points  $\mathbf{x}$  and  $\mathbf{y}$ . The kernel function can be defined as the dot product between two feature vectors.

$$k(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x})^T \phi(\mathbf{y})$$

where  $\phi(\mathbf{x})$  is a mapping from the input space (raw data) to a feature space.

If a machine learning algorithm is formulated only in terms of the dot product of two feature vectors, it this term can be exchanged with a kernel. As the kernel defines the feature space, we can work in feature space which are high- and even infinite-dimensional. This is called the kernel trick.

### 2.2.3 The Radial Basis Function

### 2.2.4 Support Vector Machines

Suppose we have data which is linearly separable. If we have only two features we can draw all samples in a 2D plot. This is shown in Figure 2.1. In this case the *best* line that can separate both classes should be as far apart from all samples as possible. This line can be defined by the samples that are nearest to it. These samples are called support vectors as they are sufficient to span the boundary. For this reason SVM is also called a sparse method as one only needs the support vectors to define the classification boundary. For higher dimensional feature spaces the same idea holds, but instead of having a line we have a plane (a hyperplane) which dissects the space in two parts. As the SVM models the boundary between each class without considering any generative process it is a discriminative model.

The assumption that the data is linearly separable can be relaxed in two ways:

Instead of finding a boundary in the feature space we can use the kernel trick to project the data into a kernel space. This way the data may not be linearly separable in the feature space, but instead could be linearly separated in some kernel space. If we take the *Radial Basis Function* for example the kernel space has infinite dimensions and thus the data can be linearly separated.

We can also allow for a small subset of samples to cross the boundary without compromising its discriminative properties. This is called the *soft-margin SVM*.

The theory behind SVM and the fact that the support vectors can be found by optimizing a convex function make this method a very robust way to do classification. For

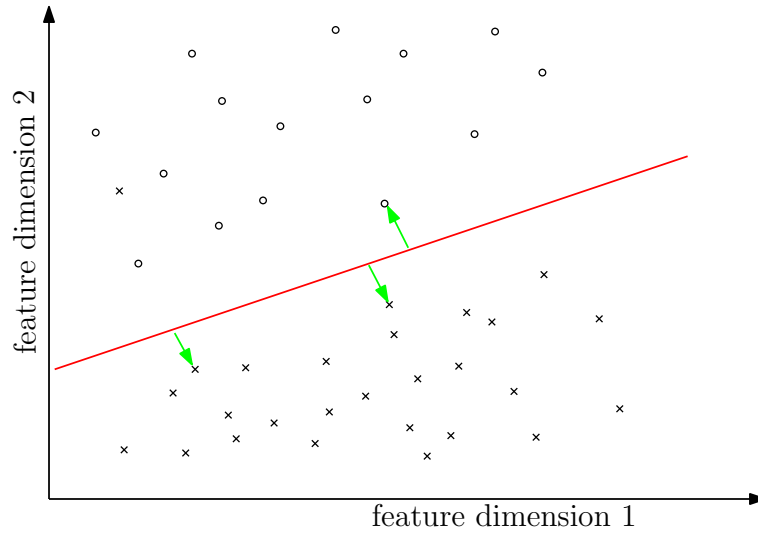


Figure 2.1: SVM decision boundary (red) between two classes (cross, circle). The support vectors are indicated in green.

this reason there are multiple implementations of SVMs which are very popular and are used very often in practical applications.

## 2.3 Gaussian Processes

Consider the multivariate Gaussian distribution above. If we want to model the distribution of a discrete function defined over a finite interval, we can treat each element of the vector  $x$  as a point of the function. Thus we can view the multivariate Gaussian distribution as a probability density function over the function space. Letting the dimensionality  $d$  go to infinity (the distance between each point goes to zero) we can model continuous functions.

In this case the mean is a point in function space, thus a function  $E[x] = f(x)$ . And because of the fact that we now have infinite dimensions the covariance can be seen as an "infinite matrix/", thus a function of two elements:  $Cov(x, y)$ . This can be also seen as a kernel as discussed in `Kernel Methods`. Therefore it can be seen as a Gaussian distribution over function space.[11]

The marginalization property is what makes Gaussian Processes feasible as it lets us compute likelihoods with a finite part of the covariance function – which can be seen as a covariance matrix.

A Gaussian process can be also seen as the bayesean posterior consisting of the product of the (Gaussian) functional prior and the observed samples. Another view is a kernelized regression with infinite parameters. [11]

A Gaussian process is a non-parametric model and is governed by the hyperparamete-

ters of the used kernel. This also means that the model is less prone to overfitting which is an important property as it not needed to perform cross validation.

### 2.3.1 The Gaussian distribution

1. **Univariate Gaussian distribution** In the one dimensional case the Gaussian distribution is well known and understood. Moreover many processes in nature can be modeled with this distribution and for this reason it is also called the Normal distribution. The probability of an event is very high on a certain "point" (its mean value  $\mu$ ) and it drops quickly on each side with the standard deviation  $\sigma$ .

$$\mathcal{N}(\mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x-\mu}{2\sigma^2}}$$

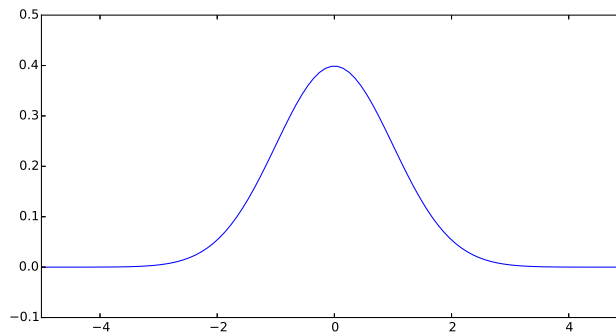


Figure 2.2: The univariate Gaussian distribution with mean  $\mu = 0$  and variance  $\sigma^2 = 1$

As you can see in Figure 2.2

One disadvantage of this distribution which we can see from the above formula is that it can model only one hypothesis. This is also the case for the Gaussian distributions of multiple (multivariate Gaussian distribution) and infinite (Gaussian process) dimensions.

2. **Multivariate Gaussian distribution** The multivariate Gaussian distribution is the generalization of the Gaussian distribution in higher dimensions.

$$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^d |\boldsymbol{\Sigma}|}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})}$$

The two parameters of the distribution are:

**mean**  $\boldsymbol{\mu} = E[\mathbf{x}]$  Representing the most probable vector



**covariance**  $\Sigma$  Representing the mutual variance for each pair of the elements of the random vector:  $\Sigma_{ij} = Cov[x_i, x_j]$

The exponent is mahalanobis distance, which measures the distance of a point to the ellipsoid defined by the covariance matrix.

3. Properties of Gaussian distributions Aside for being an appropriate model for many processes occurring in nature, Gaussian distributions are also very nice to work with. The marginal and conditional of two Gaussian distributions are also Gaussian.

One reason GPs are straightforward and work is the math behind them. It is just linear algebra operations.

Linear maps for Gaussian distributions:

Product of two multivariate Gaussian distributions:

$$\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_x, \boldsymbol{\Sigma}_x) \mathcal{N}(\mathbf{y}; \boldsymbol{\mu}_y, \boldsymbol{\Sigma}_y) = \frac{1}{\sqrt{(2\pi)^d |\boldsymbol{\Sigma}|}} e^{-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_x)^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}_x)} \frac{1}{\sqrt{(2\pi)^d |\boldsymbol{\Sigma}|}} e^{-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_y)^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}_y)}$$

Marginal of a multivariate Gaussian:

Conditional of a multivariate Gaussian:

cite  
Write  
about total  
probability  
and such

cite Gaus-  
sian Win-  
ter School  
slides  
Philipp  
Hennig  
Gaussian  
Process  
Summer  
School  
2014

### 2.3.2 Kernels

The most used kernel when using Gaussian Process is the *Radial Basis Function (RBF)*.

1. Effects of the hyper-parameters

### 2.3.3 Regression

With *Gaussian Processes* we don't learn a model, but instead we have a probability over infinitely many models with the mean being the most probable one.

### 2.3.4 Learning


In the case of a GP the learning phase is different than in parametric models, where the model parameters are inferred from the data. Training in the case of GPs means finding good hyperparameter for the kernel, by reducing the log-likelihood by variational optimization (data fit term + cov. regularizer)

In contrast to parametric models Gaussian processes are less prone to overfitting because of the covariance regularizer term.

$$E(\theta) = \frac{1}{2} \log(K) - \frac{y^T K^{-1} y}{2}$$

We see that we have to invert the covariance matrix, which is of dimensions  $n \times n$ . Therefore this operation has a runtime complexity of  $\mathcal{O}(n) = n^3$  which is also the bottleneck of the whole algorithm and a serious drawback of Gaussian Processes.

### 2.3.5 Classification

 Classifying with GPs is a little more involved, because of the discriminative function and the fact that the likelihood explain problems of GP classification right} is not a Gaussian. For this reason different models exist which try to approximate this likelihood.

### 2.3.6 Advantages

1. non parametric When using a parametric model one has to make sure that the chosen model is sufficiently complex to fit the data but at the same time is not too complex that it will overfit the training data. This is a very hard task and is in most cases done through cross-validation of the model with an independent validation set. As discussed above GPs are less prone to overfitting and therefore we do not need to reduce the training data to create a validation set.
2. probabilistic Being a model which has a Bayesian interpretation GP The hyper-parameters can be interpreted. The lengthscale controls how much neighboring points contribute to the covariance of the function.
3. generative
4. nice for Bayesian
5. linear algebra operations (marginals and conditionals)

### 2.3.7 Disadvantages

1. susceptible to outliers One big problem of the Gaussian distribution is that it has the assumption that the noise is Gaussian. When this assumption does not hold and we have several an outlier it either shift the mean un-proportionally to itself or raise the variance. Both cases are The student-t distribution, for example, is robust against outliers but is much harder to deal with.

2. Unimodal Since the Gaussian distribution is concave it can model only one hypothesis. This a curse but also a blessing since the math behind it is simple and unambiguous.
3. high computational complexity  $\mathcal{O}(n^3)$
4. non-convex optimization of the hyper-parameters

### 2.3.8 Sparse Methods

As the computation cost for inverting the covariance matrix is cubic, there are some methods which approximate the solution. One of these methods is the /Informative Vector Machine/[7] where a subset of samples is selected by maximum entropy. This way this active set can explain the data rest of the data, and using it will still result in a good model.

This reduce the complexity to  $\mathcal{O}(d^2n)$  where d is the number of chosen samples. There is also an *IVM* method which works for multiple classes.[12]

## 2.4 Gaussian Process - Latent Variable Model

The GP-LVM is an unsupervised learning model to perform a non-linear dimensionality reduction from an observed space  $X$  to a latent space  $Y$  It does this by maximizing the likelihood

$$p(Y|X) = p(Y|f)p(f|X)$$

using a Gaussian prior for the mapping  $f$ . Technically a GP-LVM is a product of Gaussian Processes which model a regression of the mapping from latent space to observed space. This means also that if we want to compute the latent position of a new observed sample we have to compute the . . . . Using a linear kernel the model generalizes to PCA. By using a non linear kernel a non-linear mapping is inferred making it a non-linear latent variable model.[6]

Analogy LVM j-ç marionettes

formulas  
etc.  
elaborate  
GP-LVM  
PCA

### 2.4.1 Dual Probabalistic PCA

Tipping and Bishop, Journal of the Royal Statistical Society (1999)

Assuming  $X$  has a Gaussian prior  $P(X) = \mathcal{N}(X|0, I)$

$$P(Y|X) = \prod_{j=1}^p \mathcal{N}(\dagger_j | \iota, \mathcal{K})$$

Where  $K = XX^T + \sigma^2 I$  is the covariance matrix. Lawrence noted that this can be interpreted as a product of gaussian processes where the covariance matrix represents the linear kernel [6]. By exchanging the linear kernel with a non-linear one, we automatically have a technique for non-linear dimensionality reduction.

Using the trace properties

$$\text{tr}(a) = a, a \text{ is a scalar}$$

$$\text{tr}(AB) = \text{tr}(BA)$$

cite some  
source for  
this

we can change the mahalanobis distance term  $x_i^T K^{-1} x_i = \text{tr}(x_i^T K^{-1} x_i) = \text{tr}(K^{-1} x_i x_i^T)$

The log likelihood is:

As there is no linear kernel any more this equation cannot be solved in closed form. Therefore we have to do gradient based optimization. But marginalizing over all latent space samples means that we have to include these in the optimization. This fact makes the optimization problem very hard as the dimensionality is high-dimensional – number of samples  $n$  + hyperparameters – and has many local minima.

As initially proposed the standard way of initializing the latent space is using *Principal Components Analysis*.

### 2.4.2 Back-constraints GP-LVM

One problem with this model is that it does not preserve local distances in the latent space. This is because it tries to explain the data by moving distant samples from the observed space also far apart in the latent space. This problem is addressed by Lawrence et al. in the back-constrained GP-LVM [8]. A mapping  $g_i(y_i) = x_i$  is introduced which constrains the points in latent space to be more near if they are also near in observed space. Instead of optimizing directly on  $X$  the back-constrained GP-LVM optimizes the mapping  $X = f(Y)$  instead. This back-constrained mapping

Having this back-constraints also gives us a mapping from observed space to latent space which can be used to project a new sample into the latent space without costly maximum likelihood estimates.

### 2.4.3 Discriminative GP-LVM

Another improvement in the context of classification in latent space is the Discriminative GP-LVM [16]. Using a *General Discriminant Analysis* criterion a prior is being enforced on the latent space which ensures that samples from one class are more clustered and different classes are more separated. This is done by maximizing the between-class

separability and minimizing the within-class variability while optimizing the log likelihood of the GP-LVM.

### 2.4.4 Other variants

1. Bayesian GP-LVM An interesting approach for computing the likelihood of the latent variable mapping was proposed in [15]. By using a variational method it becomes possible to marginalize over  $X$ . Doing so the mapping can be learned together with an ARD kernel. This way the dimensionality of the manifold can be learned from the data.

explain  
ARD

2. Subspace GP-LVM

3. Manifold Relevance Determination Combining the Subspace GP-LVM with the variational approach and the ARD kernel it is possible to learn the manifold .[4]

explain  
MRD

### 2.4.5 Advantages

1. interpolation Because of its probabilistic nature GP-LVM interpolation between two data sample is very natural. [10]
2. probabalistic
3. Generative: it can generalize beyond training data
4. non-linear mapping

### 2.4.6 Disadvantages

1. No mapping from observation space to latent space The idea of the GP-LVM is to learn a mapping from latent space to observation space by marginalization over the latent space. Resulting from this is that we do not have an inverse mapping into the latent space. This fact may be of no importance for character modeling and motion interpolation but in our case it is crucial. An inverse mapping can be computed by using the Back-constrained GP-LVM described above. However one should also keep in mind that using back-constraints inherently changes the latent space as employs an additional constraint on the mapping.
2. Very hard optimization problem Resulting from the disadvantages of Gaussian Process regarding the optimization of the hyper-parameters the GP-LVM is also very hard to optimize as its objective function is non-convex. But in the case of GP-LVM we have a much larger optimization space due to the fact the we do not optimize only the hyper-parameters, of the mapping Gaussian Process, but also the latent space itself which is of dimensionality  $n$ .

This in fact is the biggest problem as it limits its use on real world data, because for more complex manifold structures there will likely be many local minima. For this reason it is crucial to choose a good initialization. Examples are PCA, Local Linear Embedding or ISOMAP.

## 2.5 Sequence similarity measures

### 2.5.1 Dynamic Time Warping

The *Dynamic Time Warping* is an algorithm which tries to find a minimal path between two sequences where the path can be warped in the time dimension. The sequences can be of arbitrary length.

The recursive definition – excluding some corner cases – reveals the workings of this method.

$$\text{dtw}_{x,y}(i,j) = \text{dist}(x_i, y_j) + \min \begin{cases} \text{dtw}_{x,y}(i-1, j) \\ \text{dtw}_{x,y}(i, j-1) \\ \text{dtw}_{x,y}(i-1, j-1) \end{cases}$$

Where  $\text{dist}(x, y)$  is a distance function which tells how close two points are, and  $i$  and  $j$  are the element indices for the first and second sequence. The DTW can be computed with dynamic programming and has a runtime complexity of  $\mathcal{O}(nm)$  where  $n, m$  are the lengths of the two sequences.

It is closely related to the *Longest Common Subsequence* where, but instead of maximizing a common subsequence, we minimize the total warping cost between both sequences.

Since we are not interested in the path itself but in the cost of the minimal path we define the DTW as a mapping from two time series to a real value. We consider DTW to be a distance which is not entirely correct as the triangle inequality does not hold. Nevertheless it gives us a notion of how similar two time series are and since it is non-negative ( $d(x, y) \geq 0$ ), symmetric ( $d(x, y) = d(y, x)$ ) and respects the identity property ( $d(x, x) = 0$ ) it can be used to define a meaningful, but not formally correct, kernel. [13]

### 2.5.2 Longest Common Subsequence

The *Longest Common Subsequence* algorithm finds the biggest non-consecutive subsequence that is contained inside two sequences. Its recursive definition is:

$$\text{lcs}_{x,y}(i, j) = \begin{cases} \text{lcs}_{x,y}(i-1, j-1) + 1 & \text{if } x_i = y_i \\ \max(\text{lcs}_{x,y}(i-1, j), \text{lcs}_{x,y}(i, j-1)) & \text{otherwise} \end{cases}$$

This can be implemented using dynamic programming and has a run-time complexity of  $\mathcal{O}(nm)$  where  $m$  and  $n$  are the lengths of the sequences. Several algorithms exist which reduce this complexity by making some kind of assumptions about the data .

cite source  
survey  
LCS





### 3 Approach: KMeans clustering approach

As a starting point, we choose to re-implement a working method with a good performance on this data set. Therefore we choose an existing algorithm based on the *bag-of-features* approach published in 2012 [19].

The idea is illustrated in Figure `fig:bof-approach`:

- Define features which capture the structure in a time instant along with the local displacement of the skeleton. There are two types of features that are extracted. First the structural configuration is captured by the difference vector between each joint pair. Second the local motion is captured by the difference vector for frame  $t$  and  $t - 1$  for each joint. This way the feature represents the current configuration and the current motion performed for every frame. The feature vector is of size 360.
- From all poses find the most  $k$  prevalent ones. This means clustering the feature space and finding the mean vectors for each cluster. This is done by the K-Means method.
- Quantize each activity by these poses. For each activity, each frame is being mapped to a cluster mean by nearest neighbor. Doing so we have a sequence of the mean poses for each activity.
- Compute a fixed sized vector that represents the distribution of each mean pose. By computing the histogram over the previous sequence and normalizing we capture the occurrence of each pose representing the feature clusters (bag-of-features).
- Perform classification using this new feature vector. Using a linear *Support Vector Machine* we learn the activities along with their corresponding labels.

The above algorithm works very well in practice. This can be explained by the fact that the mean poses are very distinct for different activities. This means that they capture the most discriminative poses of the activity which can be robustly recognized.

Also we tested this method with only partial data and it performs relatively well. We used 100 frames uniformly sampled from each activity.

### 3.1 Cornell Daily Living Activities dataset

We will use the "Cornell Activity Datasets (CAD-60 & CAD-120)"<sup>1</sup> to learn and evaluate the performance of our implementation. This dataset is challenging as it contains complex daily living activities, some of which are very close together. There are four persons each performing 13 activities. The activities *brushing teeth*, /brushing mouth

One person in the data set is left handed and therefore the recognition ability drops considerably in this case. One way to make the method more robust for this case is to also learn the mirrored data. We do not use this approach as we wanted to compare our extensions with the original paper.

The data set consist of an sequence of frames which include:

- Image data
- RGBD data
- Skeleton information: (joint position and orientation)
- annotated meta information (e.g. activity)

### 3.2 Robot Operating System (ROS)

cite ROS

The *Robot Operating System* is a middleware which is intended to consolidate and define a layer for the implementation of complex robotic systems. It has a variety of drivers for different sensors and actors and defines a *node* based interface for communication between different sub-modules.

Each node can define a communication interface by defining message types, topics and services. This way a complex system is split in several small nodes and, because of this modularization, it is easier to add, exchange, work on and test different parts and functionality. The nodes can communicate using either pre-defined *topics* which have a message type or *services* which can also have some own defined type. A node can subscribe to a topic and each message that is then published on this topic will result in an callback.

Transforms between coordinate frames ...

A *bag* file captures all messages which are being send along with the whole topic net. This way real world data can be recorded and be played back. This is very convenient for debugging or system integration.

---

<sup>1</sup>Human Activity Detection from RGBD Images, Jaeyong Sung, Colin Ponce, Bart Selman, Ashutosh Saxena. In AAAI workshop on Pattern, Activity and Intent Recognition (PAIR), 2011.

### 3.3 Implementation

For the implementation we used Python with the `scipy` and `scikit-learn` libraries. For K-Means we used the mini-batch implementation of K-Means which is expected to perform worse than the passive variant, but also is much faster. As described in the paper we also used a linear SVM with an *RBF* kernel for classification.

### 3.4 Integration into ROS

For real time extraction of the skeleton we used the `openni_tracker` module. `/todo{cite}` This module read the values of the OpenNI nite `/todo{cite}` skeleton tracker driver and transforms the coordinates to a ros specific depth camera frame. Then it publishes these transforms as a TF message. Because of the fact that each joint TF broadcast is not synchronized we modified the module to publish the pose as an atomic message containing the skeleton positions for each frame. We also did not use any transformation, as we wanted to use the Cornell data set which is recorded with the raw data coming from the RGBD sensor. This way we could test the performance of the algorithm for online recognition without tedious creation of a new data set.

We publish the pose on the topic `/openni_tracker/pose` having the message type of an array of float32.

We implemented a new ros module called `activity_recognition` which subscribes to the above topic saves a number of poses and every three seconds performs a classification on the sequence. As the provided dataset is relatively large the learning time is several minutes. The most time takes to parse the data files and extract the features. As we did not want to do this every time we serialized a learned model and loaded it every time the module starts. This way it is also possible to learn different saved activities and begin the recognition without waiting for the model to be re-learned.

### 3.5 Shortcomings

The skeleton tracking is very noisy. We observed very big variations between subsequent frames. Therefore we performed a discrete Gaussian filter smoothing for each sequence. Unfortunately the recognition rate did not improve.

We observed that the number of prevalent poses is not sufficient to capture the variances inside some classes. For this reason we performed K-Means for each class of activities separately and used the ball-tree nearest neighbor algorithm quantize the sequences for the recognition. With this it is more likely that same activities will fall to the same representative poses as they are more evenly distributed between the classes. Moreover this allows us to extract more mean poses as the K-Means algorithm has to run only on the samples of each class separately.

The *bag-of-features* approach performs very well but it does not capture the order of

the underlying poses. Instead by performing histogram pooling, it has a notion of how prevalent each pose is for every activity.

To circumvent this we modified the method to classify with the *Longest Common Subsequence (LCS)* algorithm. Instead of performing a histogram pooling we classify each quantized sequence using the average *LCS* distance for each class. The standard algorithm for the *LCS* for two sequences is implemented, just like in the case of the *DTW*, with dynamic programming. In our case this is not needed as we already know that different activities will contain different poses. For this reason we can simply remove all elements which are not in the intersection of both sequences as a pre-processing step.

A second idea was to compare the sequences using *Dynamic Time Warping*. For this we chose as a measure between each mean pose the euclidean distance in feature space, which will give a good approximation in the case that the clusters are located far away. As the *DTW* has a complexity of  $\mathcal{O}(n*m)$  we took every fifth element from the sequence for the calculation. Also by pre-computing the distance matrix the distance operation is a simple look-up operation and the algorithm is fast enough.

One serious drawback of this approach is that only a fixed time interval can be classified. There is no way to robustly recognize transitions between different activities. For this reason we tried another approach which uses *GP-LVM* to reduce the feature space and can find the centroid for an activity in this space.

## 3.6 Evaluation

We performed 4-fold cross validation using each person as test data and the other three persons for training. We achieved a comparable precision rate of 84% and recall rate of 84% as stated in the paper. Using the *LCS* measure we achieved an precision and accuracy of 88%.

We also tested the prediction rate using only 100 frames for the prediction. By uniformly sampling 50 intervals from each test sequence the algorithm achieved the same average accuracy and precision of 88%. It can thus be argued that the most discriminative information for the classification task is inside the powerful features that are extracted and the representative poses produced by the clustering..

## 4 Approach: Discriminative Sequence Back-Constrained GP-LVM

As discussed earlier the simple *bag-of-features* approach has its limitations as it is not capable of identifying activity transitions. To deal with this problem we choose to implement another algorithm, capable of classifying a sequence in real time and inherently taking the alignment of the sequences into account.

### 4.1 Discriminative Sequence Back-Constrained GP-LVM

In the paper "Discriminative Sequence Back-Constrained GP-LVM for MOCAP Based Action Recognition"[1] the authors propose a method for classifying MOCAP actions.

cite mocap

The MOCAP database consists of a large number of different activities performed by human actors and recorded using motion capture devices. The information recorded is comparable to the skeleton representation but contains more data and is virtually noise free.

The method proposed in [1] is illustrated in Figure 4.1

The idea is to perform a dimensionality reduction on the skeleton data, resulting in a much compact representation. By introducing a *DTW* based sequence alignment kernel similarity measures can be defined for the activities. By using this similarity measure for the sequences in the observed space and constraining the optimization to preserve this measure the local distances between the sequences are transferred into the latent space. The latent points of similar sequences are thus located nearby and the centroids of similar activities are distributed more close to each other. Then instead of using back-constraints to map a single pose sample into the latent space, we can compute the centroid in the latent space directly from a sequence of poses.

The sequence back-constraints have two advantages:

First all the sequences have a meaningful clustering in the latent space and thus the mean (centroid) of each sequence is a good representation.

Second by also learning the back-constraint it is possible to calculate the centroid of a sequence in the latent space directly without maximizing a likelihood. This in turn is being used to infer the centroid for an activity in the real-time classification for actions.

The authors validated this approach on the MOCAP dataset using 7 different actions (Run, Walk, Jump, Throw-Toss, Sit-Stand, Box, Dance) and achieved an average recognition rate of 72.9%.

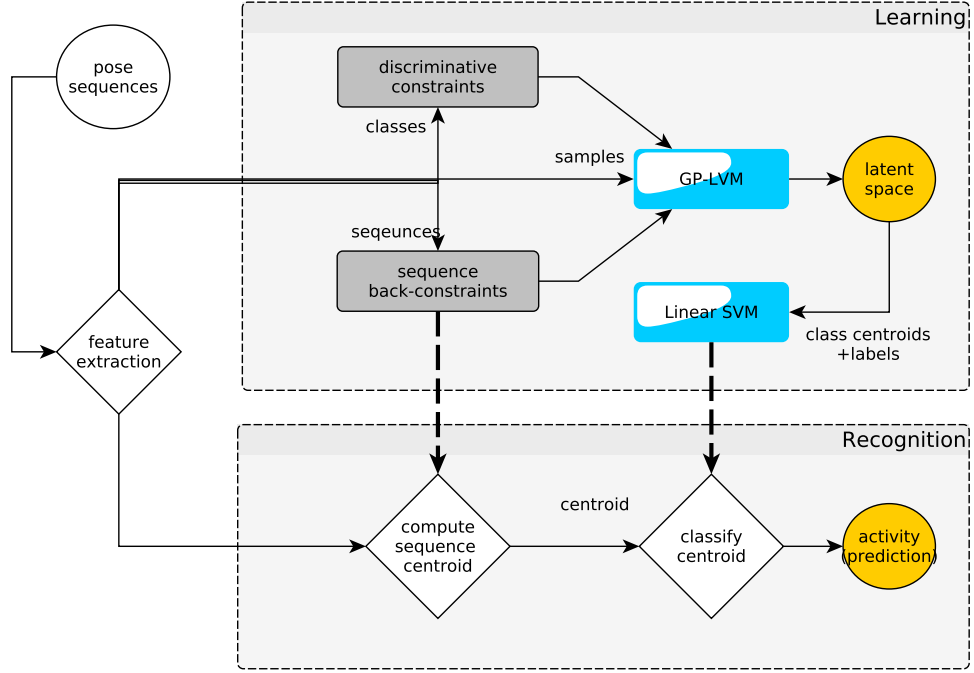


Figure 4.1: Illustration of the “Discriminative sequence back-constrained GP-LVM” approach. Learning is done by training a GP-LVM together with the sequence-back constraints and discriminative constraints. After that an Linear SVM is trained by the class centroids and the corresponding labels. Recognition is performed by computing the centroid of a new sequence through the learned back-constraints and predicting with the SVM.

#### 4.1.1 Sequence back-constraints

The mapping is defined as a linear combination of the *DTW* distance between every other sequence. For every latent dimension  $q$  we have:

$$g_q(Y_s) = \sum_{m=1}^S a_{mq} k(Y_s, Y_m)$$

where the similarity measure is  $k(Y_s, Y_m) = \gamma e^{\text{DTW}(Y_s, Y_m)}$ . This measure can be interpreted as a sequence alignment kernel. The measure is to be preserved in the latent spaces.

$$g_q(Y_s) = \mu_{sq} = \frac{1}{L_s} \sum_{n \in J_s} x_{nq}$$

Therefore we need to perform a constrained optimization for the GP-LVM.

#### 4.1.2 Discriminative GP-LVM

Furthermore, by applying the Discriminative GP-LVM we ensure that poses of different activities are separated from each other and poses from similar activities are located closer together. This ensures that the centroid of an activity is more informative and thus discriminative. The Discriminative GP-LVM works by minimizing the between class similarity and maximizing the inner-class variance [16].

The two criteria for optimization are:

- The distance between the classes

$$S_b = \sum_{i=1}^l \frac{n_i}{n} (\boldsymbol{\mu}_i - \boldsymbol{\mu})(\boldsymbol{\mu}_i - \boldsymbol{\mu})^T$$

where  $n$  is the number of samples,  $n_i$  is the number of samples for class  $i$  and  $l$  is the number of classes. Furthermore  $\boldsymbol{\mu}_i$  is the mean of the class and  $\boldsymbol{\mu}$  is the mean across all classes.

- The variance within each class

$$S_w = \frac{1}{n} \sum_{i=1}^l \sum_{j=1}^{n_i} \frac{n_i}{n} (\mathbf{x}_{i,j} - \boldsymbol{\mu}_i)(\mathbf{x}_{i,j} - \boldsymbol{\mu}_i)^T$$

where  $\mathbf{x}_{i,j}$  is the  $j$ -th sample from class  $i$ .

The variance within each class  $|S_w|$  should be minimized and the distances between the classes  $|S_b|$  should be maximized. This is done by combining both condition into a sole criterion and maximizing it:

$$J(X) = \text{tr}(S_w^{-1} S_b)$$

This criterion is added to the likelihood of the GP-LVM with a parameter  $\lambda$  which decides how much weight the discrimination should take in the optimization. If we make the model more discriminative we could break the learning of the non-linear manifold. On the other hand if the value is small the contribution will be to minor and we will not gain any discrimination between the classes. Recognition is being done by applying the mapping above to the new sequence and using a SVM in the latent space.

#### 4.1.3 Advantages

This methods maps each pose from every activity inside the same latent space, which ensures that the mapping captures a non-linear manifold which is categorized by all

activities. Recognition can be done in real time by using the learned back constrained. The centroid in the latent space is being calculated for the whole sequence and classified by the SVM. Also incomplete trajectories can be classified. When there is an activity transition the centroid will cross the decision boundary of the SVM and be naturally classified to the new corresponding activity.

##### 4.1.4 Shortcomings

As all activities are modeled inside one latent space it is very difficult to find a non-linear mapping from latent to observed space. The standard approach for optimization in the GP-LVM is using the *Scaled Conjugate Gradient* method. As the optimization for GP-LVM is determined by the above similarity measure and the discriminative criterion finding a good minimum is very difficult. It is thus highly likely that performing a gradient optimization will be stuck in an local minimum. The authors in [1] argue that initializing with a more sophisticated dimensionality reduction technique is a necessity. In their work they use the *ISOMAP* and the *Locally Linear Embedding* methods.

Also one problem with the real-time recognition is that determining when exactly an activity has ended/begun is very difficult. Also as we do not know how long a sequence is we have to calculate the centroid for several time frames using a sliding window approach.

##### 4.1.5 Extensions:

1. Learn pose together with local motion to capture dynamics The GP-LVM learns a mapping for each pose but does not consider velocities and accelerations. If we take a pose along with its first and second moments as the high-dimensional space we allow for the temporal displacements to be also modeled. The latent space will represent the pose along with the local motions and the DTW kernel in the constraint will also capture the dynamics of the activity. Due to the difficult optimization and the high complexity of the data set we could not find a good local minimum with this approach.
2. Use mahalanobis for the DTW As described in section Dynamic Time Warping with Mahalanobis Distance we wanted to use a modified version of the DTW for learning the sequence back-constraints. But due to our tests the mahalanobis inspired DTW did not perform any better for our chosen features.

## 4.2 Feature extraction

Regardless of the chosen algorithm the features used for learning will have a big impact on the performance of the model. Therefore it is imperative to extract discriminative features from the skeleton data.



We get the joint positions and the angles between them in the camera frame defined by the used depth camera (e.g. Microsoft's Kinect). When extracting features we have to make sure that we have view invariant features of the skeleton. We want these data in the frame of the skeleton.

One way to achieve scale invariance is to normalize all link lengths in respect to the torso link. This correct for variances of skeleton lengths in different persons. To make the pose view invariant we have to define a local skeleton frame which captures the skeletons *orientation* in the world coordinate system.

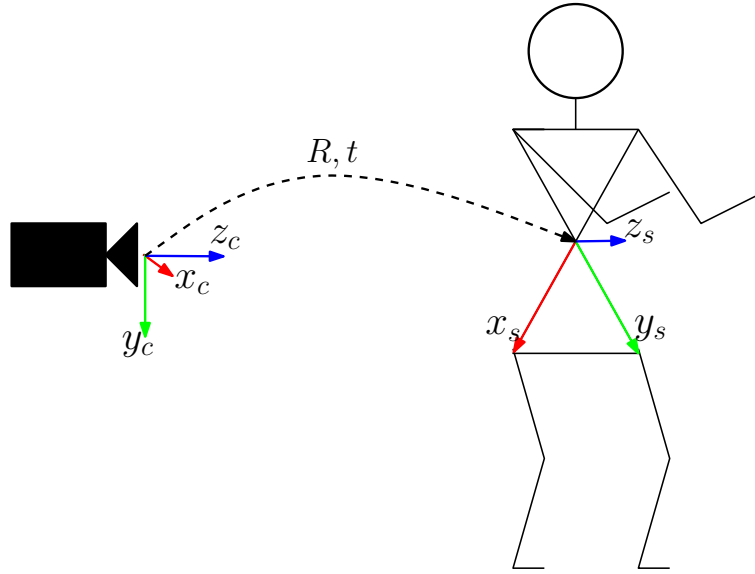


Figure 4.2: Sketch of the local skeleton frame inside the camera frame. The rotation matrix  $R$  and the translation vector  $t$  define the needed transformation to change from camera coordinates to the local skeleton coordinates

Another way to achieve view invariance is to not consider the 3D points of the joints all together but instead to take only relative features. These can be, for example the angles or distances between two adjacent joints. An interesting approach is used in [14], which is to define a polar coordinate frame for each joint and use only two angles, which define the orientation of the joint in a polar coordinate frame, as features. This way we also reduce the observation space.

As discussed in Related Work many methods also make the extracted temporal features (e.g. Eigenjoints). However since we want to include the dynamics in our model we do not extract such features explicitly.

We selected a 3D point cloud of the joints in the skeletons own coordinate frame as features. The reason for this is that we believe the 3D point cloud to be more linear than relative features, which in turn will help when optimizing the model. Figure 4.2 shows

this approach. We chose the two vectors – torso to right hip and torso to left hip – to define our local coordinate system. By normalizing and computing the cross product we have also the third vector which points to the walking direction of the skeleton.

### 4.3 Implementation

cite GPy

As there was no publicly available source code and we wanted to integrate the code with ROS we choose to implement this method in Python. We used the *GPy* library from the ... [Sheffield University](#).

To implement the Discriminative GP-LVM constraints by porting the code from Prof. Urtasun's matlab code to Python and integrating it with *GPy*.

Doing so we encountered several problems with the current numpy and scipy libraries dealing with sparse matrices. As of now there is no way to perform a fast multiplication of sparse matrices and of block-diagonal matrices. The only solution to this is by manually implementing an algorithm. But doing so one loses all the advantages of the BLAS and LAPACK integration of numpy.

To enforce the sequence back-constraints we implemented a constrained optimization by adding Lagrangians to the objective function. This way the weight parameters for each sequence alignment kernel were learned.

### 4.4 Dynamic time warping with mahalanobis distance

The Dynamic Time Warping algorithm is a prominent and very effective choice for computing similarity between two sequences. Using this measure as a sequence alignment kernel the methods aligns similar sequences closer to each other. The effectiveness of the recognition is determined by the accuracy of this alignment kernel.

The issue with this approach, in the context of activity recognition, is how to define the distance metric between two poses. This metric is crucial for the *DTW* to find an optimal path. Popular choices for the distance function is the euclidean distance, if 3D points are used as features and which we used in our implementation, and the angular distance for angles. The problem with these two distances is that they are just the sum of the individual feature differences. As the dimensionality grows this metric becomes less informative.

In the case of human poses we have a certain notion of which poses are similar and which are far apart. Maybe this is due to the fact that we inherently know – or classify – to which activity the pose corresponds to and have therefore some notion of closeness with respect to an activity which cannot be approximated with the euclidean distance. Poses from different activities will most likely also seem to be more or less similar depending on how similar the actions are.

One idea to transfer this knowledge is by using the Mahalanobis distance instead of the euclidean distance when computing the similarity of two pose sequences. By

computing the covariance for each activity we have some notion of the variance across all feature dimensions for a specific class. This way we can capture – to some extent – the variability for each class. Now we can compute a similarity measure with a new sequence  $x_{new}$  for each class and each sample of this class. Thus we can define a notion of measure between a class and a new sample by:

$$s(j, x_{new}) = \frac{1}{|C_j|} \sum_{x \in C_j} \frac{\text{DTW}_{\text{mahalanobis}(\Sigma_j^{-1})}(x, x_{new})}{\min(|x_i|, |x_{new}|)}$$

where  $C_j$  is the set containing all class sequences and  $|C_j|$  is the number of sequences in class  $j$ . The normalization factor  $\min(|x_i|, |x_{new}|)$  makes sure that the minimum cost computed by the DTW is proportional to the smallest sequence.

This way the distance error is distributed by a way defined by the variance across each dimension.

A similar idea was also proposed in the context of handwritten signature verification in [9], which uses just one covariance matrix. The covariance matrix is determined such that, just like in the case of Discriminant GP-LVM, it maximizes the variability between classes and minimizes the difference for samples in the same class. In contrast to our approach the overall covariance matrix may define a more meaningful and discriminative measure but it is also more difficult to update when performing online learning and when learning a new class (novelty detection).

#### 4.4.1 Implementation

We wrote a simple version of the Dynamic Time Warping in Python using dynamic programming and following the recursive definition in chapter 2.5.1. As the variance for some feature dimensions can be zero the constructed covariance matrix does not have full rank and thus cannot be inverted. We mitigate this problem with an approximation of the inverse by computing the pseudoinverse.

## 4.5 Evaluation

Our tests on the Cornell Daily Living Activity data were unsuccessful as the optimization failed to find a discriminative latent space. We believe that the many constraints on the optimization and the highly variant data is very hard to optimize.

Another reason for this could be that the activities in the Cornell data set are more complex. The MOCAP data represents action which could be described more easily with a non-linear manifold. In contrast most daily living activities consist of several and, also in their inherent structure, different actions. The *DTW* measure is therefore not suitable to capture the similarity between two complex activities.

It can be argued that the performance of the *GP-LVM* is strongly dependent on the initialization. This was stated also in the original [?]. It seems that the objective function

is highly-nonlinear and it is very likely that the optimization will find a local minimum in the vicinity of the starting position.

For these reasons we choose to implement a new model based on motion flow fields which will be learned for each activity separately.

## 5 Approach: GP-Latent Motion Flow

It can be argued that the mean poses computed in the *bag-of-features* method capture the most probable pose and motion tendencies of an activity. The good performance of the algorithm can be attributed to this fact. This can be explained by the local motion descriptor and the structural descriptor which are good representations for the current pose. We want to find an algorithm which performs dimensionality reduction, like in the case of the *Discriminative sequence back-constrained GP-LVM* but also captures the motion tendencies for each activity class.

Many models use GP-LVM to reduce the high dimensional space into fewer dimension. These approaches make the problem more feasible but the issue remains how to do classification for time-series data. Human motions are mostly characterized by the dynamics of the model (temporal dimension). So we have to compare trajectories in the latent space. One idea is to learn a *Gaussian Process Dynamical Model* for each activity. This way we will have a function of the trajectory. This method is very powerful but using it will very likely not give good results in our case, due to the fact that more complex activities do not necessarily resemble the same trajectory. If we take the activity "cooking" as an example, there is no main trajectory that is being followed. Moreover this activity is defined by its local motion tendencies and sub-actions. Also a trajectory based approach will not be able to model cyclic action inside an activity which can be repeated an undetermined number of times.

One idea to solve these problems is to learn a motion flow field inside the latent space. This can be with a similar method called *Gaussian Process Regression Flow (GPRF)* proposed in [5]. The classification can be done using first and second order dynamics which should give better results. Going further the activity itself is characterized by the first and second moments of the trajectory function. By explicitly modeling the velocity of the trajectory we can take changes in the joint movement into account.

### 5.1 Gaussian Process Regression Flow

[?] can be used to model the trajectories in the latent space.

explain  
GPRF

## 5.2 GP-Latent Motion Flow

The GP-LMF method is inspired by this model. The difference being that in the case of activity recognition we do not know the starting position and also the trajectories can have significantly different lengths. For this reason it is very difficult to normalize with respect to the time dimension. Nevertheless, resulting from the properties of Gaussian Process regression, we have also a dense mean flow field and dense variances. This allows us perform efficient and robust online recognition in the latent space.

This model is attractive for two reasons. First real-time classification of incomplete trajectories is possible. Incomplete not only in the sense of the first part of an activity but any interval of an activity, which could be also somewhere in the middle of the sequence. Second it is possible to do online learning by simply adding the new class as a new flow field to the pool of GP regressions. It is very difficult to adjust other models, which rely on the mapping between latent space to observation space, for online learning, because of the problem that we can get stuck in a local minimum when optimizing the parameters of the GP-LVM.

The idea is to learn a motion field in the latent space for each activity. This can be achieved by learning the velocity function of the latent point just like in the GPRF model presented above. With the difference that we do not use the spatio-temporal domain but only the spatial domain for the latent space. The reason being that we do not have starting and ending positions for each activity and also the lengths can be variable. On top of that we also want to recognize an activity which is being interrupted by another activity, so we cannot fix the lengths of the trajectories.

Each activity has its own flow field. Recognition and prediction is done by calculating the energy of the currently moving point with each different field. The field with the minimum energy represents the most probable activity as the point follows more closely its "current" of motion.

Variances in the speed of performing an activity can be modeled by giving the point in the latent space a mass which can be adjusted in real time. When a point has greater mass then it needs more energy to be propagated through the flow field (the overall activity is slower) and vice versa.

As we use the GP-LVM and a GP regression way we have two indicators for recognizing unobserved data. The first one is the variance of the back-constraint mapping. If it is high we know that current sample is far apart from the observed ones. The second is the variance of the *Gaussian Process Regression*. If this value is high we know that we didn't see any sample in the latent space with the current motion. Therefore, with this two indicators, we have a notion of how new a sample and its current motion are.

Another advantage of this method is that activities with repetitive motions, such as walking or running, can be learned without using periodic kernels or without resorting to model them explicitly. Repetitive motions can be seen as just multiple samples of the same motion which define the flow field.

Figure 5.1 illustrates this approach.

- For every activity class the latent space representation is learned.
- Then we compute the numerical derivative of the sequence inside latent space.
- 

### 5.3 Requirement: Smooth latent space

It is imperative for the *Gaussian Process* regression that the latent space is smooth. If this is not the case then different poses which are located nearby can have completely different gradients. As the *GP-LVM* preserves distances rather than locality, it is very likely that there will be no smooth mapping in the latent space. One approach to constrain the latent space to be smooth is using the back-constraints.

### 5.4 Learning the flow field

The initial idea was to learn a general dimensionality reduction for a high number of varying activities and work with only one latent space. But as we saw in Chapter 4 such an approach is very difficult to realize, because of the difficult optimization task. Another problem is that it is very difficult to learn a smooth mapping in the latent space. This is described more deeply in [18] where the authors try to incorporate the optimization criterion of *Locally Linear Embedding* together with the a back-constrained *Gaussian Process Dynamical Model*. As this approach needs also prior knowledge and is very complex we decided to learn each activity separately. Future work should deal with the possibilities of learning a unified latent space as it will allow us to learn different flow fields in the same space and we will not have to perform a heuristic normalization of the different flow fields.

One problem we encounter by learning the motion flow field from several samples is complexity of the *Gaussian Process*. There are two solutions for this. The first one is to use a sparse GP model. The second one is to sample points from all samples and use only those that are most suitable for the regression. If we take *IVM* as the sparse GP model both approaches can be seen as equivalent as the *IVM* will automatically take the most informative samples.

#### 5.4.1 Effects of the hyperparameters

Changing the *lengthscale* defines how much each point is contributing to the regression process. It can be interpreted as a smoothness factor which governs how strong the interpolation of the flow field is performed on the latent points.

Changing the signal variance controls how much

## 5.5 Interpretation

The proposed model has a natural interpretation. A point represents a pose in latent space and an activity is a trajectory in time inside the same space. With the flow field we learn the motion tendencies for each pose. When performing recognition we let the current point traverse each separate flow and compute the needed energy. If we consider that the point has a mass we can model the speed at which activities are being performed. This way we can recognize when a point leaves an activity, which represents a *motion current*, and passes over to some other activity.

The model captures the changes in velocity which is comparable to the motion history images discussed in Chapter `Related Work`.

## 5.6 Recognition

The first approach for computing the energy for a new sequence and each flow field was to compute the dot product of the actual gradient with the vector from the flow field at the latent point mapped from the back-constrained mapping. This quantity is then added to the weight of the particle.

Inspired by the particle filter method our recognition approach was to have a particle for the latent space of each activity. In every time step the particle is being updated with the above described probability. Then all particles are normalized. This way we ensure that the particle represents the probability that the current action is being performed. If it respects the flow field it will accumulate more weight and due to the normalization the other particles will become smaller.

## 5.7 Advantages

### 5.7.1 Recognition

The current activity is being mapped into the latent space. Through the learned back-constrained. The recognition is being performed solely in the latent space. By propagating the current position by each flow field we can calculate the next possible pose. By comparing the similarity considering the variances we have a measure of how well the current activity resembles each flow field e.g. learned activity. By doing this for each separate activity class we can maintain a probability how likely it is that the current motion tendencies resemble each learned class.

### 5.7.2 Prediction

If we have detected the activity predicting is simply a matter of propagating the pose through the flow field by taking the mean of the GP. This way short-term predictions



are possible. Also the prediction is updated every time the point changes its position in the latent space.

### 5.7.3 Multiple Hypothesis Prediction

Since we have a GP representing our flow field we can predict future point positions with the mean value. Moreover also having informative variances we can sample several possible trajectories. This can be accomplished using a particle filter. Hence we can have multi-hypothesis predictions along with their probabilities.

### 5.7.4 Online learning

### 5.7.5 Active learning

### 5.7.6 Natural interpretation

### 5.7.7 Novelty detection (anomaly detection)

In [?] the authors present the ability of the GPRF model for anomaly detection. This approach is also suitable for finding new classes as the above energy value can be used to recognize novel activities. The reasoning is that if we cannot find a flow field with a small energy the activity has to be unobserved.

### 5.7.8 In comparison to the GPDM it can model cyclic activities

## 5.8 Implementation

We used non-sparse *GP-LVM* with a back-constrained mapping to learn each class of activities. The back-constrained has two purposes. The first is that we need a backward mapping to map new poses to the latent space. The second is that the latent space must be smooth.

## 5.9 Problems

### 5.9.1 Dimensionality reduction

Performing a non-linear dimensionality reduction is no easy task. Testing was done with only two dimensions as it is easier to visualize the latent space and the resulting flow fields. A latent space with higher dimension will naturally make the reduction more robust and the field will have a more natural interpretation. . . .

### 5.9.2 Stable class mean flow field

When learning a stable flow field from several samples the field can degenerate with the inclusion of strong variable paths. Therefore it is important to ensure that the algorithm

active  
learning  
- problem  
??

learns stable paths. This can be achieved by sampling uniform random sampling from all samples of the same activity.

## 5.10 Evaluation

At first we concentrated our efforts for learning with the MOCAP data. In theory the data collected from the kinect should be equivalent. One difference is the high noise in the pose estimation, but due to the fact that the GP-LVM preserves distances rather than locality this problem is mitigated to a certain degree. For most activities, such as walking, running and jumping it was possible to learn a representative flow field. For more complex actions the optimization could not find a smooth mapping. This could also be due to the used *RBF* kernel for the backward mapping.

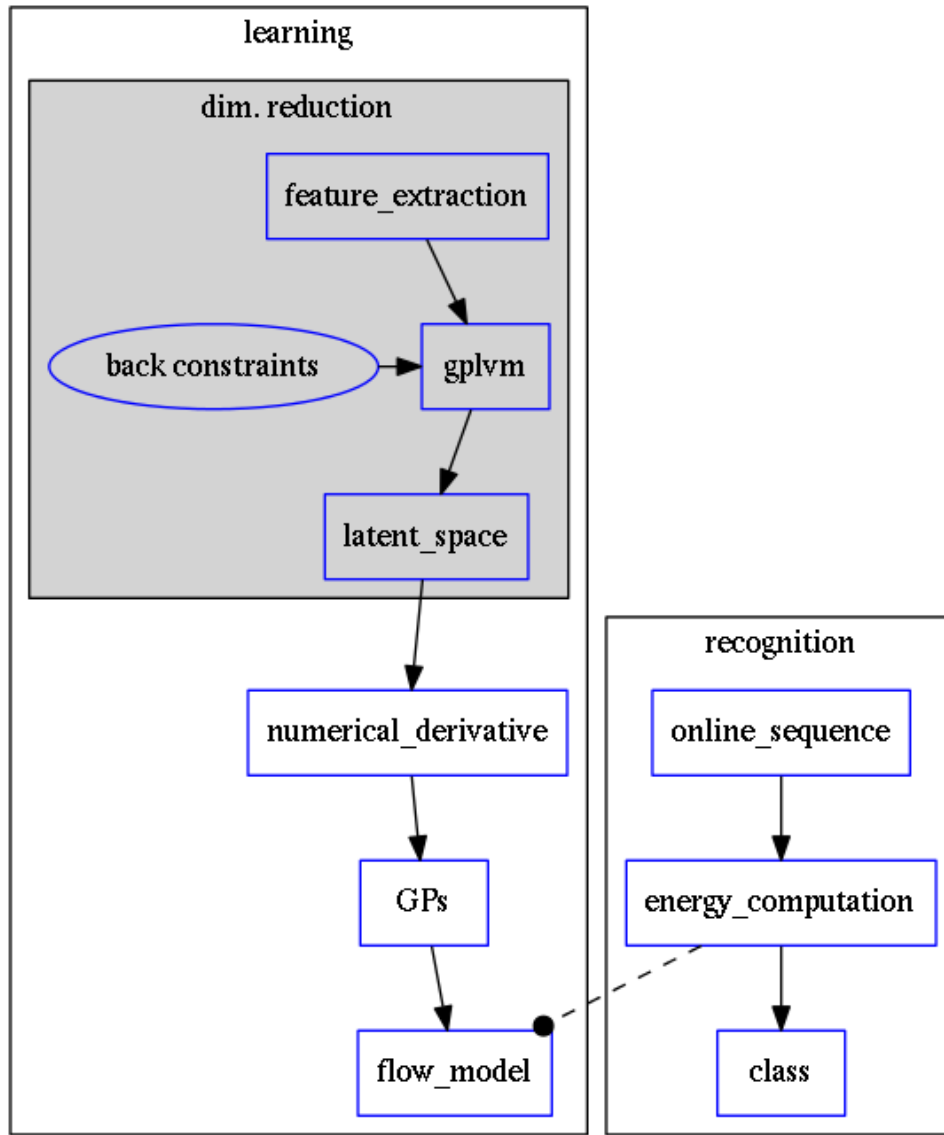
Unfortunately we were not able to perform an appropriate dimensionality reduction. This several reasons:

- Due to the fact that we want to also learn a backward mapping (observed space to latent space) we have to initialize the latent space with this mapping. For this reason we could not define a more appropriate initialization such as *PCA* or *LLE*.
- The Cornell dataset contains many activities with many samples. Using the *IVM* variant helps in some respect to circumvent this problem but also introduces an approximation.

[18] [3] [17]

The author in Exploring model selection techniques for nonlinear dimensionality reduction also suggest to use ISOMAP or LLE to initialize the GPLVM and argues that direct optimization of the GP-LVM is very difficult.

Probabilistic Feature Extraction from Multivariate Time Series using Spatio-Temporal Constraints



Gaussian Process - Latent Motion Flow

Figure 5.1: Illustration of the "Gaussian Process - Latent Motion Flow Field" approach.



## 6 Conclusions and Outlook

### 6.1 Summary

In this thesis we covered the issue of performing online activity recognition from skeletal features. We began by re-implementing an existing method, which works by extracting representative poses and performs a histogram pooling over the quantized pose sequence.

### 6.2 Lessons learned

**6.2.1 Dimensionality reduction for all activities is very difficult (also with extra constraints)**

**6.2.2 Dynamics is a good measure for classification of human activities**

### 6.3 Contributions

**6.3.1 Implementation of**

**6.3.2 Advantages and Disadvantages of dimensionality reduction with GP-LVM for human motion in the context of activity recognition**

**6.3.3 Implementation of the Discriminative GP-LVM with python**

We ported the matlab code provided by Prof. Urtasun into python and integrated it with the GPy library

**6.3.4 Implementation of the Sequence Back-constraints**

We used Lagrangians to implement a constrained optimization of the likelihood function

**6.3.5 Improvement of the DTW measure with the mahalanobis distance**

**6.3.6 A novel approach for activity recognition**

With the *GP-LMF* we presented a novel method for online recognition of complex activities.

### 6.3.7 Introduction of an energy minimization approach for online recognition of complex activities

## 6.4 Outlook

### 6.4.1 Implementation of the GP-LMF using spatio-temporal GP-LVM

cite spatio  
temporal  
GP-LVM

As described earlier the GP-LMF approach failed, due to the fact that the optimization of the GP-LVM with back-constrained did not result in a smooth latent space. One possibility to solve this problem is by implementing the *Spatio-Temporal GP-LVM* as described in .

### 6.4.2 Semi-supervised activity learning by automatic segmentation of activities

If the problem of a smooth latent space can be solved the *GP-LMF* method can be used to perform an automatic segmentation of observed and unobserved activities. As discussed earlier, since we have two good indicators of the uncertainty of both, the poses we see and the motion tendencies, we can segment a time interval into "known" and "unknown" activities. In conjunction with online learning, this will greatly reduce the time this method needs to learn appropriate flow fields for a number of classes.

## List of Figures

2.1	SVM decision boundary (red) between two classes (cross, circle). The support vectors are indicated in green. . . . .	9
2.2	The univariate Gaussian distribution with mean $\mu = 0$ and variance $\sigma^2 = 1$	10
4.1	Illustration of the "Discriminative sequence back-constrained GP-LVM" approach. Learning is done by training a GP-LVM together with the sequence-back constraints and discriminative constraints. After that an Linear SVM is trained by the class centroids and the corresponding labels. Recognition is performed by computing the centroid of a new sequence through the learned back-constraints and predicting with the SVM.	24
4.2	Sketch of the local skeleton frame inside the camera frame. The rotation matrix $\mathbf{R}$ and the translation vector $\mathbf{t}$ define the needed transformation to change from camera coordinates to the local skeleton coordinates . . .	27
5.1	Illustration of the "Gaussian Process - Latent Motion Flow Field" approach.	37





# Bibliography

- [1] Discriminative sequence back-constrained GP-LVM for MOCAP based action recognition:. pages 87–96. SciTePress - Science and and Technology Publications, 2013.
- [2] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, August 2006.
- [3] Sebastian Bitzer and Christopher Williams. Kick-starting GPLVM optimization via a connection to metric MDS. 2011.
- [4] Andreas Damianou, Carl Ek, Michalis Titsias, and Neil Lawrence. Manifold relevance determination. *arXiv preprint arXiv:1206.4610*, 2012.
- [5] Kihwan Kim, Dongryeol Lee, and Irfan Essa. Gaussian process regression flow for analysis of motion trajectories. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 1164–1171. IEEE, 2011.
- [6] Neil Lawrence. Probabilistic non-linear principal component analysis with gaussian process latent variable models. *The Journal of Machine Learning Research*, 6:1783–1816, 2005.
- [7] Neil Lawrence, Matthias Seeger, and Ralf Herbrich. Fast sparse gaussian process methods: The informative vector machine. *Advances in neural information processing systems*, pages 625–632, 2003.
- [8] Neil D. Lawrence and Joaquin Quiñonero-Candela. Local distance preservation in the GP-LVM through back constraints. In *Proceedings of the 23rd international conference on Machine learning*, pages 513–520. ACM, 2006.
- [9] Yu Qiao, Xingxing Wang, and Chunjing Xu. Learning mahalanobis distance for DTW based online signature verification. In *Information and Automation (ICIA), 2011 IEEE International Conference on*, pages 333–338. IEEE, 2011.
- [10] Sébastien Quirion, Chantale Duchesne, Denis Laurendeau, and Mario Marchand. Comparing GPLVM approaches for dimensionality reduction in character animation. 2008.
- [11] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. University Press Group Limited, January 2006.

- [12] Matthias Seeger and Michael I. Jordan. Sparse gaussian process classification with multiple classes. Technical report, Citeseer, 2004.
- [13] Hiroshi Shimodaira, Ken-ichi Noma, Mitsuru Nakai, and Shigeki Sagayama. Dynamic time-alignment kernel in support vector machine. In *NIPS*, volume 14, pages 921–928, 2001.
- [14] Ilias Theodorakopoulos, Dimitris Kastaniotis, George Economou, and Spiros Fotopoulos. Pose-based human action recognition via sparse representation in dissimilarity space. *Journal of Visual Communication and Image Representation*, 25(1):12–23, January 2014.
- [15] Michalis Titsias and Neil Lawrence. Bayesian gaussian process latent variable model. 2010.
- [16] Raquel Urtasun and Trevor Darrell. Discriminative gaussian process latent variable model for classification. In *Proceedings of the 24th international conference on Machine learning*, pages 927–934. ACM, 2007.
- [17] Raquel Urtasun, David J. Fleet, and Pascal Fua. 3d people tracking with gaussian process dynamical models. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 238–245. IEEE, 2006.
- [18] Raquel Urtasun, David J. Fleet, and Neil D. Lawrence. Modeling human locomotion with topologically constrained latent variable models. In *Human Motion—Understanding, Modeling, Capture and Animation*, pages 104–118. Springer, 2007.
- [19] Chenyang Zhang and Yingli Tian. RGB-d camera-based daily living activity recognition. *Journal of Computer Vision and Image Processing*, 2(4):12, 2012.