Learning and Generalizing Robot Behaviours using Invariant Representations

handed in
Interdisciplinary Project

Roc Reguant

born on the 04.03.1991
living in:
Paradiesstr. 3
80538 Munich
Tel.: 0179 3612830

Chair of
AUTOMATIC CONTROL ENGINEERING
Technical University of Munich

Univ.-Prof. Dr.-Ing./Univ. Tokio Martin Buss
Univ.-Prof. Dr.-Ing. Dirk Wollherr

Supervisor:	M.Sc. Matteo Saveriano
Start:	01.12.2015
Intermediate Report:	22.09.2016
Delivery:	27.10.2016

TODO: In your final hardback copy, replace this page with the signed exercise sheet.

TODO: Before modifying this document, READ THE INSTRUCTIONS AND GUIDELINES!

Abstract

In a fixed environment, robots are able to perform narrowly defined tasks satisfactorily. Robots without defined environments face serious troubles due to the complexity of the environment. All kinds of environment conditions cannot be programmed beforehand because it is impossible to predict and encode all the possible tasks. In these cases, robots must be able to learn each task from people with limited technical background. To solve this problem, experts introduced the learn by demonstration concept. Learning by demonstration uses probabilistic machine learning algorithms to extract and exploit regularities of each task. In this report we generalize from multiple trajectories. Firstly, we transform the trajectories from the 3D Cartesian space into the DHB invariant space. The transformation from cartesian to invariant return the invariant motion and the inital frame that enables us to reconstruct from the invariant trajectory to the original trajectory. Instead of using the reconstruction frame we use the identity matrix to reconstruct the trajectory. With the identity matrix replacing the reconstruction frame we obtain 3D Cartesian trajectories that share the same space. For the generalization Gaussian Process and a Gaussian Mixture Model have been used to learn the trajectory. Finally, we multiply the learned trajectory with the first frame from any desired trajectory to obtain the desired prespective for the reconstructed motion. Data has shown this approach to be successful in taking advantage of the invariant space properties without its drawbacks. During the second part of this report we learned to predict the velocity based on the current position. The velocity predicton enabled us to predict the next position assuming that the position in time $t + 1$ is a derivative of the current position plus the velocity.

Contents

Chapter 1

Introduction

With the reduced size of electronics, robots can be exported from industrial complexes (where they traditionally have been placed) to any kind of environment such as our home. Domestic robots should be able to perform the tasks that we are not able or have no time to perform. In Japan, for example, with its ageing population, robots are replacing humans in human-care centers such as hospitals. Even if we are physically able to perform the tasks, we might not always find the time.

In comparison to industrial robots, domestic robots are designed to coexist with humans in a changing environment; therefore domestic robots should have the flexibility to handle changes so they can adapt themselves to the surroundings. Due to the dynamically changing environment, domestic robots should be able to generate dynamically new trajectories from their previously learned trajectories [Sch06]. Storing infinite movement motions for every possible circumstance is impossible. Moreover, humanoids are designed to be in people's homes where no programming skills are expected; therefore robots should be able to learn new movements from the user's representations.

Learning from motion trajectories, as well as their generalization to different scenarios, is important to increase the robot's versatility in everyday scenarios. With this purpose, in this study we have implemented a method to learn from multiple invariant trajectories. The invariant enables us to obtain the pure movement independent from the point of view allowing us to generaliza better. The invariants are encoded using the DHB invariant space [SSL15]. We also compared the generalization in the cartesian space.

1.1 Related Work

Many research has been conducted to generalize from multiple trajectories in the Cartesian space and from a single trajectory in other invariant spaces [CGB07]. Until now there is no research that successfully generalize from multiple DHB invariant

representations. Different research used distinct statistical techniques to exploit regularities of certain tasks [SG00]. Gaussian Processes (GP) is a popular choice to exploit regularities from trajectories that has been successful in other experiments [DFR15]. It is able to deal with uncertainty and perturbation. In research such as [KF09], they acknowledge the importance of using Gaussian Processes along with trajectories in order to improve tracking models. The advantages of GP are that they consider both noise in the system and uncertainty in the model. Moreover, with decrease in the certainty, the model degrades gracefully. The drawback of GP is its computational complexity, $\mathcal{O}(n^3)$ [KF11]. Gaussian Mixture Models are a mixture distribution providing a combination of probability distributions [Rey09]. Since Gaussian Mixture Model (GMM) uses a probability density approach, it is noise tolerat and performs good with unobserved parameters. It embedds segmentation, so it might not provide an accurate representation [Rey09] [Do08].

Different publications show the advantages of using invariant spaces for gesture recognition. An invariant space was proposed by [SL13] which is a coordinate free space composed of two invariants. This two coordinate invariant space is not bidirectional (once converted can not be transformed back to the Cartesian space). The inability to transform back the invariant data into Cartesian data makes this space suboptimal. Another invariant space is presented in [DS10] but it is too sensitive to noise (shown in [SSL15]). Both of this problems are solved by DHB invariant space [SSL15] which represent motion trajectories in 3 invariant axis. Using DHB invariant space the conversions between cartesian and invariant spaces can be performed in both directions without any information loss while reducing noise sensitivity.

Chapter 2

Proposed Approach

2.1 Implementation

Many approaches have been developed to generalize trajectories in the Cartesian space. The problem of generalizing based in the Cartesian space is that different motions difering on the point of view require a preprocessing with matrix transformations or other methods to be able to generalize from it. Aiming to solve the Cartesian space problems, several researches have developed different invariant spaces [SL13] [DS10] [SSL15]. The invariant spaces are spaces where all the motions are encoded in a similar way independent from the Cartesian space. Until the now, many invariant spaces were only uni-directional meaning that once the motion was converted to invariant could not be recovered back to Cartesian. In this report we propose to generalize from multiple invariant trajectories and be able to reconstruct the cartesian representation. To achieve the generalization, we transformed the cartesian motions into DHB invariant representations. During the research we tried different approaches to generalize from the DHB invariant. The successful attempt has been to not generalize from the invariant space but from the reconstructed space. The reconstructed motion is obtained using the identity matrix to force all the representations to be maped into the same space. Applying the desired frame to the generalization produces an accurate Cartesian trajectory that represents the learned motions.

2.1.1 DHB invariant space

We used DHB invariant space to transform the Cartesian trajectories into invariants. DHB invariant space representation simplifies gesture recognition in real scenarios. For example, different people might perform the same gesture or different points of view might be involved. Contrary to affine transformations this method does not require fixed points in the image plane [Rie86]. Other representations do not include orientation or are not able to retrieve the Cartesian data but DHB invariant does [SSL15]. An advantage of using DHB invariant space is that orientation is no

longer a problem since all the trajectories would be encoded in the same way [SSL15].

This invariant representation consist of three values, one for the velocity and the other two for the rotation. DHB invariant space introduces a three time instant sample delay because it requires the current time instant and the next two (three in total) to compute the invariant [SSL15].

The frame that describes how the position changes over time is composed by three vectors (which is the minimum representation) [SSL15].

The x-axis is a unitary vector with the difference of the two consecutive positions.

$$\hat{x}_p(t) = \frac{\Delta p(t)}{||\Delta p(t)||}$$

The y-axis is the normal between the x-axis at the current instant t and the x-axis at the next instant.

$$\hat{y}_p(t) = \frac{\hat{x}_p(t) \times \hat{x}_p(t + \Delta t)}{||\hat{x}_p(t) \times \hat{x}_p(t + \Delta t)||}$$

The z-axis is the product of the two previous axis (x-axis and y-axis).

$$\hat{z}_p(t) = \hat{x}_p(t) \times \hat{y}_p(t)$$

Once we constructed the frame, the invariant representations can be defined. To describe the motion of the body, the following formula has been used [SSL15]

$$m_p(t) = ||\Delta p(t)|| = \Delta(t) \cdot \hat{x}_p(t)$$

And then to align $\hat{x}_p(t)$ to $\hat{x}_p(t + \Delta(t))$ the angle θ_p^1 needs to be rotated $\hat{y}_p(t + \Delta(t))$ and to align $\hat{y}_p(t)$ to $\hat{y}_p(t + \Delta(t))$ then θ_p^2 needs to be rotated about $\hat{x}_p(t + \Delta t)$. This rotation invariant angles are defined as [SSL15]

$$\theta_p^1(t) = arctan\left(\frac{\hat{x}(t) \times \hat{x}_p(t + \Delta t)}{\hat{x}_p(t) \cdot \hat{x}_p(t + \Delta t)}\right) \cdot \hat{y}_p(t)$$

$$\theta_p^2(t) = arctan\left(\frac{\hat{y}(t) \times \hat{y}_p(t + \Delta t)}{\hat{y}_p(t) \cdot \hat{y}_p(t + \Delta t)}\right) \cdot \hat{x}_p(t)$$

Notice that both formulas are formally the same but they differ on the affected axis.

2.1.2 Gaussian Processes

We used Gaussian Processes (GP) to generalize the motions. GP is a key aspect of our research because it optimizes expensive functions assuming that there is an underlying single function with noise. It is a good method because it can learn from

data that has sensor noise or in which the points are dispersed. When the data sets are large, GP is unpractical due to the cubic complexity of the training [Ras06]. The Gaussian Process concept is

$$\hat{y}_j^S = f(x_j) + \epsilon$$

where $\hat{y}_j^S$ is the observed value and $f(x_j)$ is the function on the point x_j plus an accumulated error of ϵ. The accumulated error is encoded as $N(0, \sigma_n^2)$ and represents the random noise variable which is independent and identically distributed. Gauss' error theory states that accumulating small and independent errors lead to a Normal distribution [Ras06].

We used the Radial Basis Function (RBF) kernel to compute the underlying function. This kernel, also called squared exponential, is used to compute the covariance function $k(x_i, x_j)$. Covariant functions assume that close input points will have similar target values $\hat{y}_j^S$. We chose this kernel because it has already been proven to be a good kernel for such tasks [BCDS08]. RBF representation is

$$k(x_i, x_j) = \sigma_f^2 exp\left(-\frac{(x_i - x_j)^2}{2l^2}\right)$$

where $(x_i - x_j)$ is the euclidean distance between points and the function variance is represented by σ_f^2. The length scale l smooths to a greater or lesser degree the prediction determining how well the mean matches the training data.

To prepare for the Gaussian Process regression firstly the covariance function is calculated summarizing all the points in three matrices.

$$K = \begin{bmatrix} k(x_1, x_1) & k(x_1, x_2) & ... & k(x_1, x_n) \\ k(x_2, x_1) & k(x_2, x_2) & ... & k(x_2, x_n) \\ ... & ... & ... & ... \\ k(x_n, x_1) & k(x_n, x_2) & ... & k(x_n, x_n) \end{bmatrix}$$

$$K_* = \begin{bmatrix} k(x_*, x_1) & k(x_*, x_2) & ... & k(x_*, x_n) \end{bmatrix}$$

$$K_{**} = k(x_*, x_*)$$

The Gaussian Process Regression learn the predictive distribution $p(f_*|x_*)$ of a new test output f_* given a test input x_* [SE10]. The best estimate for x_* is $\hat{y}_*^S$ which is the mean of the distribution and obtaining it using the following matrix multiplication [Ebd08].

$$\hat{y}_*^S = K_* K^{-1} \mathcal{Y}$$

The variable $\mathcal{Y}$ here represents the data set. The other variables can be seen in previous descriptions.

The uncertainty is modeled through the variance is also computed using the previous matrices.

$$var(x_*) = K_{**} - K_* K^{-1} K_*^T$$

2.1.3 Gaussian Mixture Models

Contrary to GP, Gaussian Mixture Models (GMM) use the expectation-maximization algorithm (EM). It is an iterative algorithm that converges the data points to the local maximum [VVK03]. Expectation-Maximization algorithm is a method that finds the parameters' maximum-likelihood (ML) of an underlying distribution from a given dataset with non-observed variables (missing or incomplete) [B$^+$98]. The algorithm iterates between two steps. It starts with the E-step, or expectation step, which estimates the missing points based on the existing data (the observed and the already estimated points). After the E-step, the M-step re-estimates the parameters assuming that the results previously computed by the E-step were right [NH98]. To get the final result EM algorithm repeats this two steps, E-step and M-step, until convergence. The convergence is assured because the model improves increasing the likelihood at each iteration [DB08] [Bor04].

To determine the optimal number of clusters required for the model Bayesian information criterion (BIC) has been used for this experiment. This approach has been selected because it has performed well in several applications [BSP$^+$04] [CB08] [HGCB08]. BIC adds a penalty to the log-likelihood based on the number of parameters [FRMS12] and its formula is

$$\text{BIC} = -2\log(\hat{L}) + k\,\log(n)$$

where n is the number of data points in the model and k is the number of free parameters to be estimated in the model. $\hat{L}$ is the maximized value of the likelihood function of the model and is decoded as $\hat{L} = p(x|\hat{\theta})$ where x is the observed data and $\hat{\theta}$ are the parameters of the model. We iterated creating GMMs with different cluster number checking whether its BIC was worse than the previous one. Once the BIC value was worse than the previous one the algorithm gets to the point where more clusters do not improve the classification; therefore the previous number of clusters is the correct one.

As stated by [Rey09] GMM is a summation of weighted densities represented by the following formula

$$p(\xi_j) = \sum_{k=1}^{K} \pi_i \mathcal{N}(x_j; \mu_k, \Sigma_k)$$

where ξ_j is the data point which is defined by the probability density function. The number of clusters in the model is represented by K and each cluster has a weight associated represented by π, the component centers are described by μ and the covariance matrix are encoded as Σ. The covariance matrix can be splitted as [CB08]

$$\Sigma_k = \begin{pmatrix} \Sigma_k^{TT} & \Sigma_k^{TS} \\ \Sigma_k^{ST} & \Sigma_k^{SS} \end{pmatrix}$$

Then for each component there is an expected ξ_j^S for a given x_* defined as

$$p(\xi_j^S | x_*, k) = \mathcal{N}(\xi_j^S; \hat{\xi}_j^S, \hat{\Sigma}_k^{SS})$$

$$\hat{\xi}_j^S = \mu_k^S + \Sigma_k^{ST}(\Sigma_k^{TT})^{-1}(x_* - \mu_k^T)$$

$$\hat{\Sigma}_k^{SS} = \Sigma_k^{SS} - \Sigma_k^{ST}(\Sigma_k^{TT})^{-1}\Sigma_k^{TS}$$

The GMM distribution is defined by

$$p(\xi_j^S | x_*) = \sum_{k=1}^{K} \beta_{k,j} \mathcal{N}(\xi_j^S; \hat{\xi}_k^S, \hat{\Sigma}_k^{SS})$$

The $\beta_{k,j} = p(k|x_*)$ is the probability of a component k to be responsible for x_* and is computed as

$$\beta_{k,j} = \frac{\pi_k \mathcal{N}(x_*; \mu_k^T, \Sigma_k^{TT})}{\sum_i^K \pi_i \mathcal{N}(x_*; \mu_i^T, \Sigma_i^{TT})}$$

The estimation of the conditional expectation of $\hat{\xi}_j^S$ given x_* is defined by $p(\xi_j^S | x_*)\, \mathcal{N}(\hat{\xi}_j^S, \hat{\Sigma}_k^{SS})$ uses the linear transformation property of Gaussian distributions [CB08] whose parameters are

$$\hat{\xi}_j^S = \sum_{k=1}^{K} \beta_{k,j} \hat{\xi}_k^S$$

$$\hat{\Sigma}_j^{SS} = \sum_{k=1}^{K} \beta_{k,j}^2 \hat{\Sigma}_k^{SS}$$

By evaluating both $\hat{\xi}_j^S$ and $\hat{\Sigma}_j^{SS}$ in the sample time interval $x_* \in [0, T]$ (where T in our data is 57 for the invariant space and 60 for the Cartesian space) we obtain the generalized trajectory and its variance extrapolated from the training data.

Chapter 3

Experimental Results

3.1 Learning from one invariant

We started learning one invariant and reproducing it. Figures 3.1, 3.2 and 3.3 are the representations of letter M in the DHB invariant space. The blue line is the predicted trajectory and the green dots are the data samples used for training. GP is used to learn the motion and build the model. As the figures disply, the predicted data matches perfectly the training data.

We reconstruct the invariant motion to a 3D Cartesian motion using the same frame for the invariant trajectory and predicted one. Figures 3.4, 3.5 and 3.6 show that the reconstruction was successful only using one sample as a training for the model. The model matched successfully the trajectory to all three axis. Since the Figures 3.1, 3.2 and 3.3 showed that the matching was perfect we can forecast that the reconstruction would be also a perfect match.

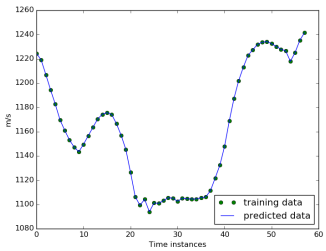

Figure 3.1: Velocity from the learned invariant of letter M

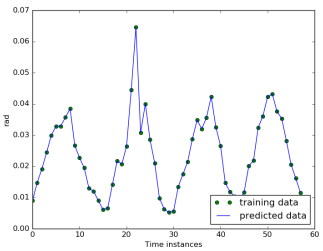

Figure 3.2: θ^1 from the learned invariant of letter M

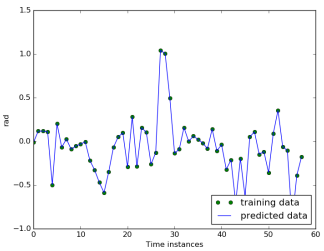

Figure 3.3: θ^2 from the learned invariant of letter M

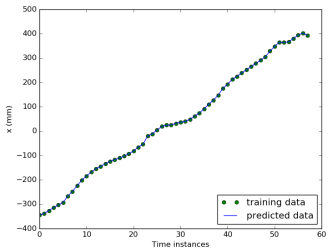

Figure 3.4: X-axis from the reconstruction of letter M using one invariant

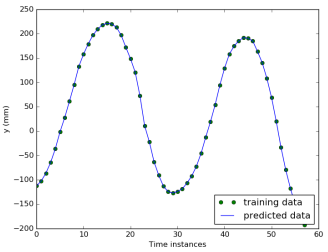

Figure 3.5: Y-axis from the reconstruction of letter M using one invariant

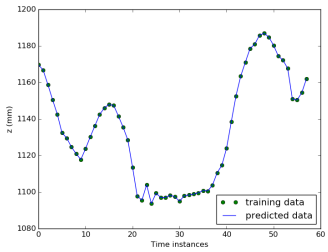

Figure 3.6: Z-axis from the reconstruction of letter M using one invariant

One sample as a training data set does not always provide a good generalization. Applying the same concept over a different letter, such as letter A provide generalization problems due to the extreme deviations in some data points. The outlier points found in Figure 3.9 for the θ^2 axis cause a bad generalization that produces an accumulated error in the reconstruction. Figures 3.7, 3.8 and 3.9 show that GP is not able to successfully predict all the trajectories in the invariant space as accurately as it was done for letter M. GP does not consider the outliers to be valid points since they differ by several standard deviations from the mean. Figures 3.10, 3.11 and 3.12 show that the reconstructed trajectory differs from the original trajectory because of the bad predicted trajectory. The reproduction is precise until a non-accurate prediction is obtained. Then the reconstruction starts suffering from the expected values. The reconstruction starts deviating. Because of the nature of the DHB invariant space each value is the velocity increment or the rotation increment respect the previous position; therefore if at some point the rotation does not meet the expected values the deviation increases with the time. One wrong angle leads to an accumulated error that increases the deviation over time.

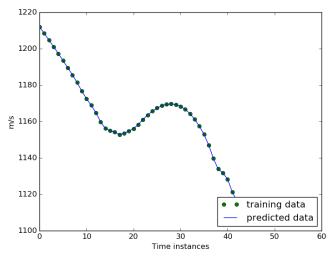

Figure 3.7: Velocity of the learned invariant from letter A

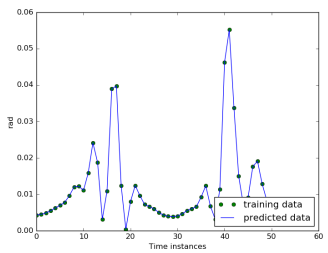

Figure 3.8: θ^1 of the learned invariant from letter A

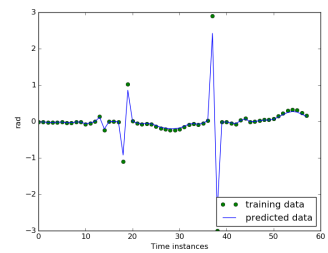

Figure 3.9: θ^2 of the learned invariant from letter A

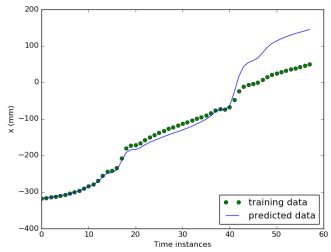

Figure 3.10: X-axis from the reconstruction of letter A using one invariant

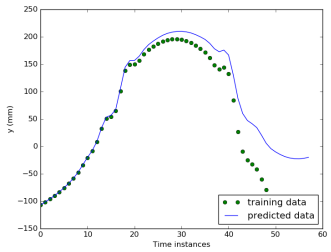

Figure 3.11: Y-axis from the reconstruction of letter A using one invariant

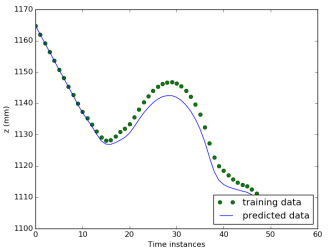

Figure 3.12: Z-axis from the reconstruction of letter A using one invariant

3.2 Learning from two invariants

The aim of this report is to generalize from multiple invariants as a first step we tried to learn from two invariants. Figures 3.13, 3.14, and 3.15 show a good invariant space generalization. The predicted trajectory in all figures is accurate. The problem arises when the reconstruction is done. Figures 3.16, 3.17 and 3.18 show that the rotations are the principal cause of problems in the reproduction. The figures show a similarity in shape but the reproduction show several degrees of deviation. These degrees of deviaton increase the distance with the time because the movement of the trajectory is applied in the wrong direction.

To prevent deviations in the trajecotry, we used the rotations of the training samples instead of learning the rotations. So, we trained the model only based on the velocity. Figures 3.19, 3.20, and 3.21 show that if we train the model only with the velocity and use the corresponding rotations for the reconstruction, the prediction of our model produces more accurate results. The results have significantly improved but this is not a real generalization. To generalize we cannot learn from one axis and use the exact same training data to reproduce the other two axis.

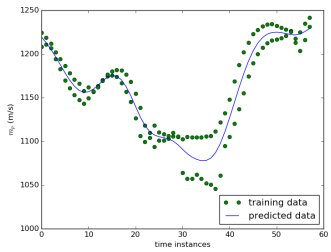

Figure 3.13: Learned velocity invariant from letter M

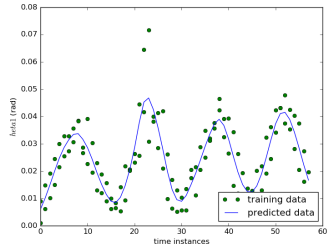

Figure 3.14: Learned θ^1 invariant from letter M

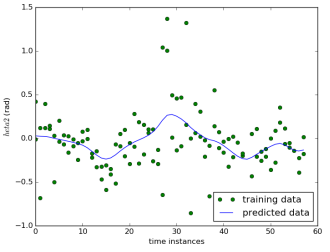

Figure 3.15: Learned θ^2 invariant from letter M

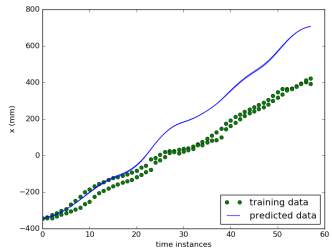

Figure 3.16: Learned X-axis from two invariants for letter M

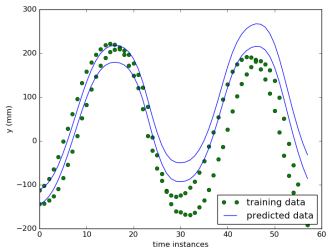

Figure 3.17: Learned Y-axis from two invariants for letter M

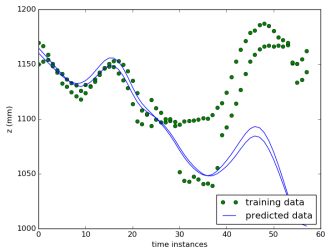

Figure 3.18: Learned Z-axis from two invariants for letter M

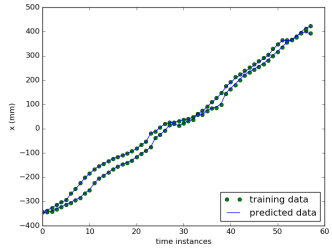

Figure 3.19: X-axis from the partially learned motion

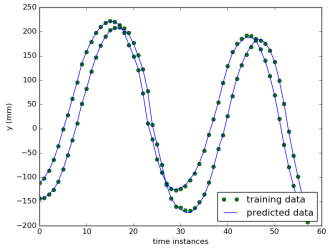

Figure 3.20: Y-axis from the partially learned motion

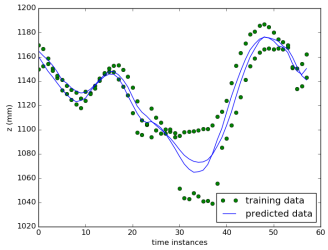

Figure 3.21: Z-axis from the partially learned motion

3.3 Learning from the accumulated invariant values

We assumed that the high data variability in the invariant space could cause generalization problems. We assumed that the sudden trajectory changes in the invariant space did not help the GP to produce a fitting model. We thought that GP returned too normalized predictions. To prevent this we accumulated the values obained in the DHB invaiant space so that there were no extreme values and the GP accurately could learn the regularities. Previous plots shown that the data points are highly irregular, contrary Figures 3.22, 3.22, and 3.22 show that the data is highly concentrated even in some cases the predicted data overlaps the training data. In this case the data dispersion is smaller and the patterns are easily spottable. The GP found a better generalized trajectory.
Eventhough we successfully showed that GP learns the motion with ease. We did not succeed to provide a generalization in the Cartesian space. Figures 3.25, 3.26, and 3.27 show that the reconstruction was not successful at all. The basic motion can be appreciated but the reconstruction of the predicted trajectory has little reasable to the trained motions.

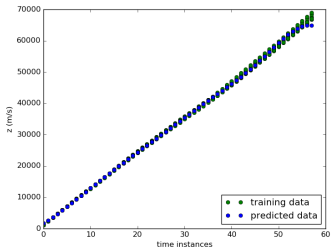

Figure 3.22: Accumulated velocity axis

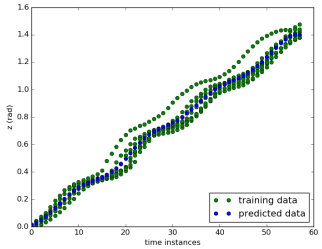

Figure 3.23: Accumulated θ^1 axis

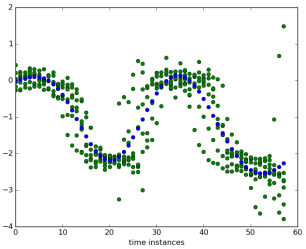

Figure 3.24: Accumulated θ^2 axis

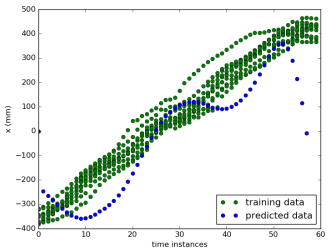

Figure 3.25: X-axis from the learned motion

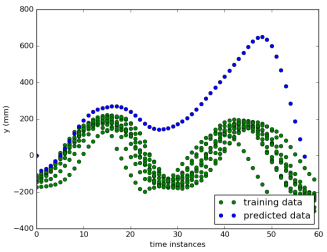

Figure 3.26: Y-axis from the learned motion

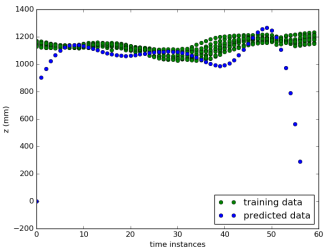

Figure 3.27: Z-axis from the learned motion

3.4 Learning from the invariants and trying to predict the reconstructed trajectories

The last unsuccessful approach has been to give the invariants as input to the GP and train the model with the reconstructed trajectories as output. The Figures 3.28, 3.29 and 3.30 show that it is not possible to successfully generalize because there is too much data dispersity. The problem arrises when we appreciate that there are many points concentrated in a specific area. If the concentrated points want to be generalized we are loosing precision assuming that the deviation is noise which is in fact not true. Now with time as X-axis the Figure 3.31 seem to be able to perform a good generalization since the blue line (predicted data) and the red dots (expected values) do follow a similar pattern. We can see in Figures 3.32 and 3.33 that the generalization perform the worst in θ axis achieving an extreme level of smoothness not matching the reality.

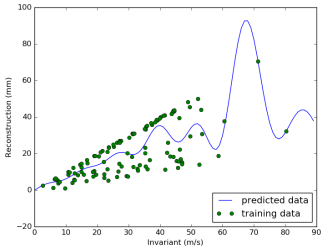

Figure 3.28: Velocity axis as input and reconstruction as output

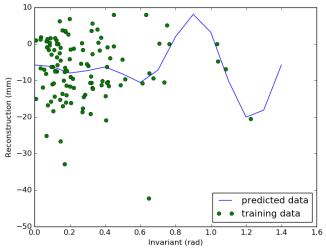

Figure 3.29: θ^1 axis as input and reconstruction as output

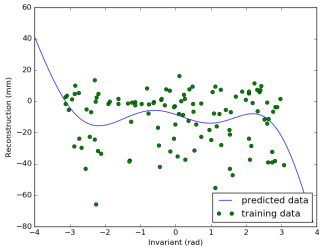

Figure 3.30: θ^2 axis as input and reconstruction as output

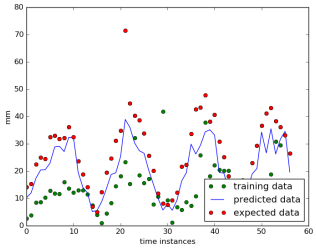

Figure 3.31: Velocity axis as input and reconstruction as output

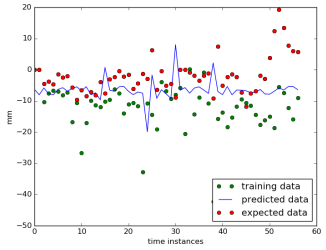

Figure 3.32: θ^1 axis as input and reconstruction as output

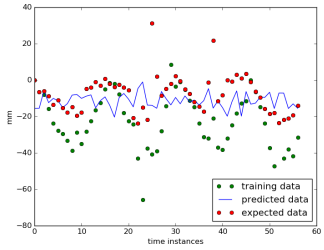

Figure 3.33: θ^2 axis as input and reconstruction as output

3.5 Generalizing from the reconstructed invariants

We successfully generalized a motion using the properties that DHB space offers. To successfully generalize a motion we followed few simple steps shown in Figure 3.34. The main property affected by DHB is the point of view independence. The same motion recorded from different prespectives are encoded in the same way in the DHB space. We firstly transformed all the Cartesian trajectories to DHB invariant trajectories to obtain the standadrized motion. The DHB algorithm transforms the Cartesian trajectory to an invariant trajectory and returns a reconstruction frame. This frame is used for the DHB algorithm to successfully reconstruct the invariant into the original 3D Cartesian motion. For this experiment, we reconstructed the invariant trajectories using the identity matrix, not the reconstruction frame. We used the identity matrix to represent all the motions in the same plane at its "purest" form, meaning without any additional transformation. Since the trajectories were firstly encoded in the invariant space they had all the properties that the invariant space has. The point of view did not have any influence in the reconstruction since

we used the same generic frame, the identity matrix, for all the motions. With the identity matrix we made sure that the provided result was the least altered representation of every invariant motion.
Figures 3.35, 3.36, and 3.37 are the reconstructions of the trajectories using the identity matrix. The green dots are the training trajectories and the blue line is the generalized trajectory. In this case, GP is able to successfully generalize the trajectories since they are reconstructed again into the Cartesian space but using the same frame for all of the reconstructions. As it is displayed the GP generalizes the motion adapting the function at each point. The generalization can be transformed to a real motion with a valid prespective using an initial frame. The reproduction can be seen in Figures 3.38, 3.39, and 3.40. The figures follow the same coloring pattern like the previous ones plus a red trajectory which is the motion that provided the initial frame to build the reconstructed trajectory. The trajectory represented in red was not used for the training. We did not use the red trajectory because it would have caused conflicts to the experiment quality. This generalization follows the training samples closely. The most distant points are in the Y-axis. The reconstruction can be closer depending on the initial frame.
The generalizations using GMM are found in Figures 3.41, 3.42, and 3.43. This figures also disply a good generalization but in this case we can see the segmentation that is caused by the intrinsic nature of GMM. The generalization is more linear. Since we are working on linear motions this generalization performs even better than GP. The reconstruction with GMM is displied in Figures 3.44, 3.45, and 3.46. As it can be apreciated, this reconstructions have successfully approximate the used motions. In fact we can see that the reconstructions using the initial frame are more accurate with GMM than with GP.

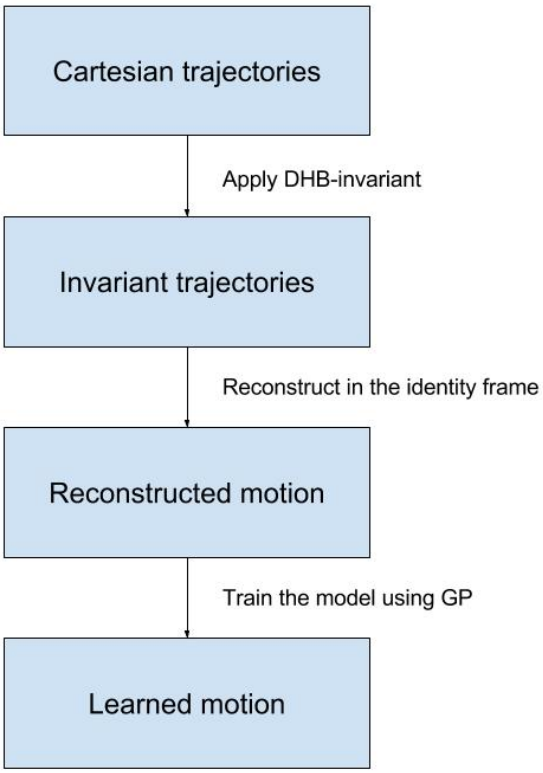

Figure 3.34: Generalization work flow

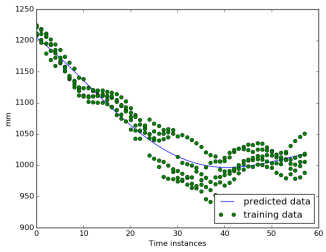

Figure 3.35: Reconstructed X-axis using identity the matrix generalized with GP

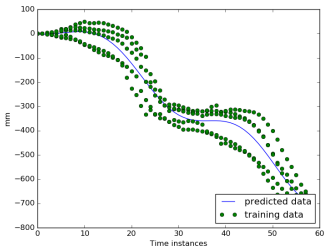

Figure 3.36: Reconstructed Y-axis using identity the matrix generalized with GP

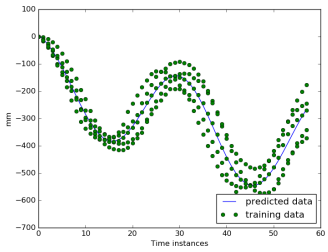

Figure 3.37: Reconstructed Z-axis using identity the matrix generalized with GP

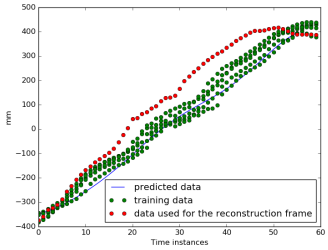

Figure 3.38: Reconstructed X-axis using their respective initial frame generalized with GP

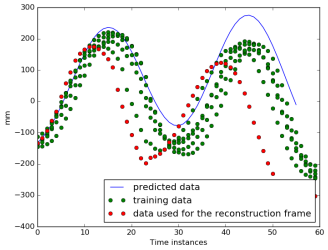

Figure 3.39: Reconstructed Y-axis using their respective initial frame generalized with GP

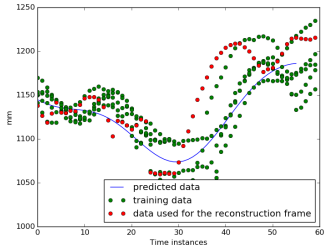

Figure 3.40: Reconstructed Z-axis using their respective initial frame generalized with GP

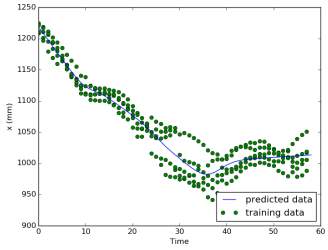

Figure 3.41: Reconstructed X-axis using identity the matrix generalized with GMM

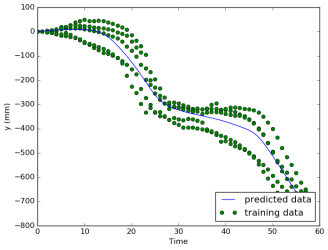

Figure 3.42: Reconstructed Y-axis using identity the matrix generalized with GMM

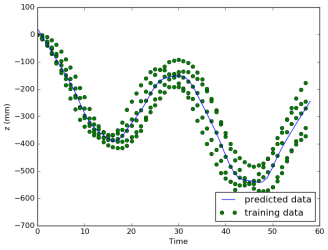

Figure 3.43: Reconstructed Z-axis using identity the matrix generalized with GMM

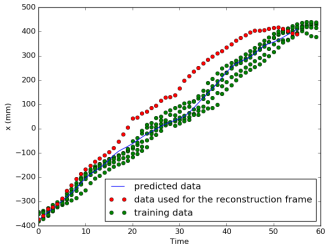

Figure 3.44: Reconstructed X-axis using their respective initial frame generalized with GMM

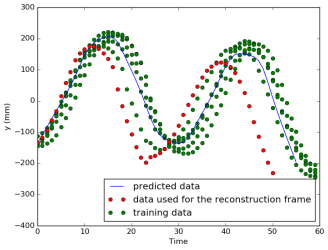

Figure 3.45: Reconstructed Y-axis using their respective initial frame generalized with GMM

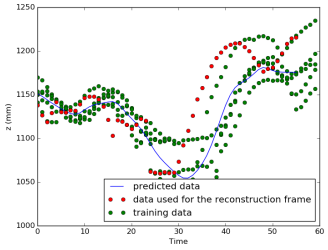

Figure 3.46: Reconstructed Z-axis using their respective initial frame generalized with GMM

We used Mean Squared Error (MSE) to estimate the accuracy of the predictions and add tangible data to this work. The MSE evaluation method is a simple estimator

which computes, as the name indicates, the mean $\frac{1}{n}\sum_{i=1}^{n}$ of the squared errors $(\hat{Y}_i - Y_i)$ between the testing points and the predicted values [KPPB07]:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)$$

We use this evaluation metric to test which algorithm approximates better the predicted values. The lower the number the better the algorithm. This metric only allows comparisons among identical measurement units. Mixing axis would cause a distortion of the result. Even using the same units, different axis might provide results in a different order of magnitude. Different order of magnitude cause that the axis with the biggest order of magnitude has a bigger impact on this metric.

As we can see on Table 3.1 the GMM algorithm approximates better the trajectories in the invariant space than GP. This is because almost all letters are compose of straight lines. Those straight lines are better represented by GMM since the regression of GMM between clusters is percieved as an stright line. GMM regression was favoured by the underlying segmentation. Contrary to GP that provides a more smooth line depending on the points near the desired data point.

Tables 3.2 and 3.3 represent the data obtained using GP and GMM from the mesurements in the cartesian space. The first table represents the motion learned only in the Cartesian space and the second table represents the motion learned in the invariant space but transformed into the Cartesian space. The data show that the trajectories learned in the Cartesian space generalize better than the ones learned in the invariant space and then transformed into the Cartesian space. In all cases the trajectory learned directly in the Cartesian space generalizes better than the one learned in the invariant space. This is caused for two reasons. The first reason is that we are learning in a different space and when the generalization is transformed into the Cartesian space the informaion has changed. The second reason is that since we learned the invariant space we require a frame to transform the trajectory to the Cartesian space. There is no universal frame for the reconstruction. Each trajectory produces its own first frame; therefore depending on the trajectory's first frame used the reconstruction can vary significantly. Even though the comparasions are done in the cartesian space the frame has a big impact on the final result.

The importance of using the right frame can be seen in Figures 3.47, 3.48, and 3.49. The green dots represent the motion as we captured it and the red dots represent the same motion but reconstructed using a different trajectory frame. As it can be seen the reconstructions differ significantly. This is one of the factors why the reconstructions from the learned motions in the invariant space might perform worst than the learned motions directly in the Cartesian space. The fact that a learned invariant generalizes better from different prespectives compensates the

Table 3.1: MSE in the DHB invariant space

		A	M	N	X	O	Avg
GP	m_v [mm/s]	8.82	16.64	18.13	22.71	31.10	19.48
	θ^1 [rad/s]	34.55	61.92	75.48	87.10	94.33	70.67
	θ^2 [rad/s]	62.35	101.10	145.06	155.05	161.84	125.08
GMM	m_v [mm/s]	8.78	13.98	18.38	22.07	31.57	**18.96**
	θ^1 [rad/s]	26.31	46.02	73.55	68.50	67.69	**56.41**
	θ^2 [rad/s]	28.68	40.40	68.73	71.39	68.22	**55.48**

Table 3.2: MSE for motions learned in the Cartesian space

		A	M	N	X	O	Avg
GP	X [mm]	16.56	19.99	22.37	31.17	57.55	29.53
	Y [mm]	19.55	27.53	32.70	24.21	63.55	33.51
	Z [mm]	8.29	17.21	14.28	22.18	33.51	19.09
GMM	X [mm]	16.46	19.15	23.07	33.41	37.71	**25.96**
	Y [mm]	19.60	30.32	36.08	26.27	36.64	**29.78**
	Z [mm]	5.61	14.17	15.14	20.05	17.61	**14.52**

generalization problems attributed to the first frame.

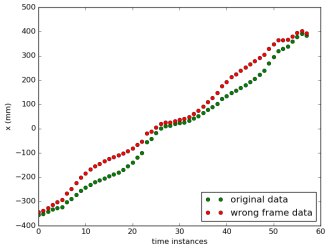

Figure 3.47: Reconstruction with inverted frames for axis X

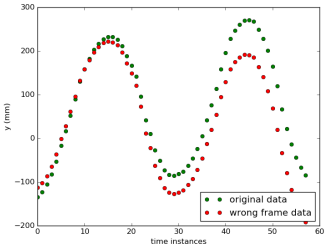

Figure 3.48: Reconstruction with inverted frames for axis Y

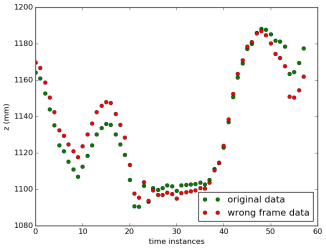

Figure 3.49: Reconstruction with inverted frames for axis Z

3.6 Learn position-velocity relationship

We predicted velocity based on the position because it enables us to continue the motion at any point of the trajectory. It makes it easier to predict the next steps. If we use time to reproduce the motion we could only reproduce the trajectory in a time linear motion. If we try to predict the velocity based on the current data point the trained GP works accurately. The resemblance between the original data and the predicted is high. Figues 3.50, 3.51, and 3.52 show that we can successfully predict the next position. We did not predict the exact next position but the velocity in a

Table 3.3: MSE for motions learned in the DHB space but transformed into the cartesian space

		A	M	N	X	O	Avg
GP	X [mm]	16.16	30.36	24.91	110.80	66.35	49.72
	Y [mm]	27.28	62.35	35.95	106.96	66.76	**59.86**
	Z [mm]	5.89	18.16	14.39	24.54	23.20	17.24
GMM	X [mm]	22.69	19.98	68.07	51.89	63.41	**45.21**
	Y [mm]	40.62	33.06	86.03	54.44	126.73	68.18
	Z [mm]	7.91	14.19	15.32	20.66	26.38	**16.89**

given point. Since the current position is known and we can predict the velocity, we can successfully predict the next position. Figures 3.53, 3.54, and 3.55 show the non trained trajectory used for the reproduction and the predicted positions. The figures disply high degree of similarity between the predicted values and the trajectory used for the prediction. Tables 3.4 and 3.5 show that the invariant and cartesian generalization values using MSE.

To demonstrate the properties of the invariant, we translated and rotated a non-trained trajectory to show that we can successfully generalize from different prespectives and positions. Figures 3.56 and 3.57 show the training data in green and in red the data used to predict the velocity. As it can be seen they are the same trajectory but depending on the case is translated to the right or rotated 90°. Figures 3.58, 3.59, and 3.60 show the prediction accuracy. For the last plots the red dots are the real trajectory positions and the blue line is the predicted next position using the current known position plus the predicted velocity for the axis. Notice that we are predicting the position $t + 1$. All three figures display almost a perfect match between predicted position and real position.

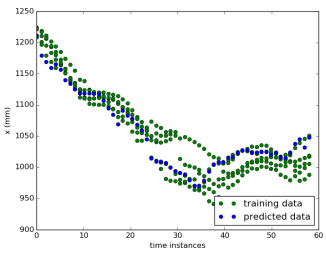

Figure 3.50: Predict velocity for the X-axis based on a identity matrix reconstruction

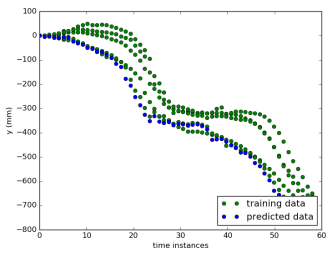

Figure 3.51: Predict velocity for the Y-axis based on a identity matrix reconstruction

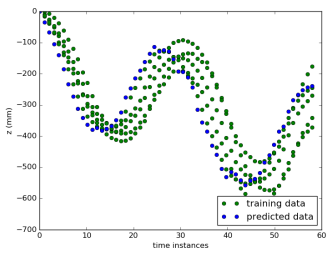

Figure 3.52: Predict velocity for the Z-axis based on a identity matrix reconstruction

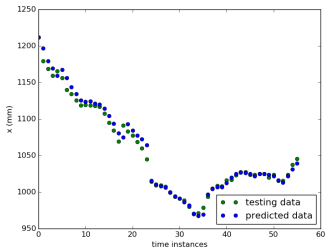

Figure 3.53: Predict velocity for the X-axis based on the Cartesian reconstruction

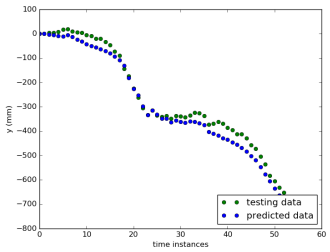

Figure 3.54: Predict velocity for the Y-axis based on the Cartesian reconstruction

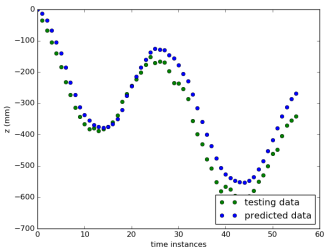

Figure 3.55: Predict velocity for the Z-axis based on the Cartesian reconstruction

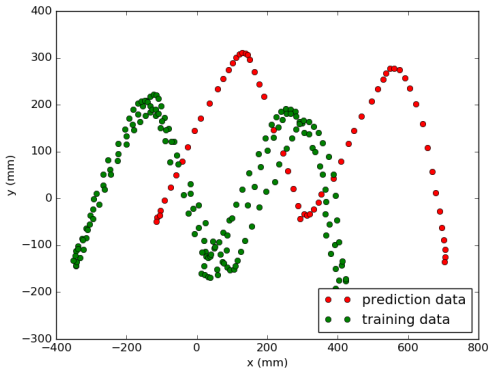

Figure 3.56: Training trajectories and the one used to predict velocity translated

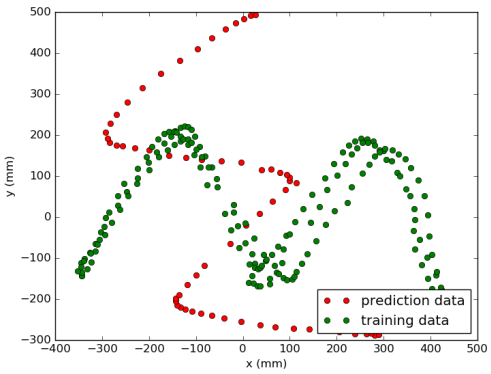

Figure 3.57: Training trajectories and the one used to predict velocity rotated

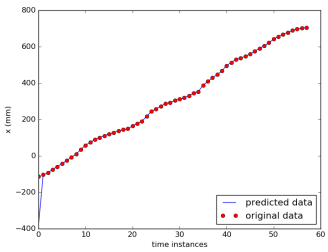

Figure 3.58: Real trajectory and predicted next position on the X axis

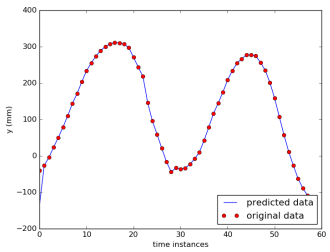

Figure 3.59: Real trajectory and predicted next position on the Y axis

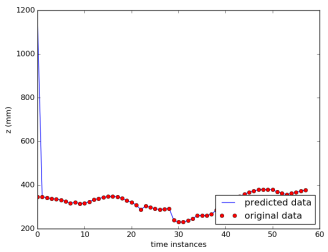

Figure 3.60: Real trajectory and predicted next position on the Z axis

Table 3.4: MSE for the predicted velocity and the real position in the invariant space

		A	M	N	X	O	Avg
GP	m_v [mm/s]	3.36	5.82	6.34	2.44	13.25	6.24
	θ^1 [rad/s]	13.59	48.66	53.82	13.89	97.94	45.58
	θ^2 [rad/s]	10.57	33.89	46.75	19.05	49.95	32.04

Table 3.5: MSE for the predicted velocity and the real position in the Cartesian space

		A	M	N	X	O	Avg
GP	X [mm]	3.36	5.95	6.52	2.64	13.36	6.36
	Y [mm]	13.59	14.23	19.96	14.45	18.68	16.18
	Z [mm]	10.57	21.16	16.79	20.20	24.85	18.72

Chapter 4

Conclusion

The goal of this research was to successfully generalize from multiple invariants. Until now, we tried to train the model using the invariant trajectories and failed to do so. The reproduction from the learned invariants turned out to be unsuccessful because the predicted trajectory substantially differed from the samples. The problem of learning from the invariant is that the wrong rotation lead to a minimal difference that over time produces a substantive deviation from the original motion. The deviaton accumulates over the time because the movement in the worng direction leads to an increased distance. The best working generalization is done through the reconstructions that used the identity matrix instead of the initial frame. We first encoded the trajectories into the DHB invariant space to obtain similar representations from the motions independently from the point of view. When the trajectories are converted from the Cartesian space to the invariant space the algorithm returns the invariant trajectory and the initial frame. The initial frame is used to reconstruct the trajectry in order to recover the original motion. Instead of using the original frame we used the identity matrix to reconstruct the trajectories. The use of the identity matrix allowed us to project all the motions into the same space. These reconstructed trajectories enabled us to take advantage of all the invariant space properties without falling for the generalization problems that we had if we learned from the invariant space. This space does not lead to an accumulated error as it happens if we learn from the DHB invariant space. It also prevented us from the problem of learning in the cartesian space since all the motions have the same prespective. We used GP and GMM to generalize the motions. GP displied a smoothness and higher adaptability to the whole motion adapting itself to small variations in the training data. GMM, due to the segmentation, offered a less nosy data and better results in most of the cases. Once we learned and generalized from the reconstructed identity trajectories, we can apply any frame to obtain the motion in the desired plane. In this case we used a frame from a non-trained trajectory to see how the trajectory looks like and confirm that the predicted trajectory has a close resemble to both the trained and non-trained trajectories.

The second goal of this research has been to predict the velocity of each axis based on the 3D position. This goal was successfully achieved with a high accuracy. Successfully predicting the velocity enables the algorithm to predict the next position of our trajectory. We are able to predict the next position of our trajectory because it is the derivative of the current position plus the velocity which we forecasted. This enables us to reproduce the motion starting from any point. Since we learned the velocity using the invariant the prediction can be done for any trajectory independently of the prespective.

Appendix A

Extra plots

A.1 Generalizing from the reconstructed invariants

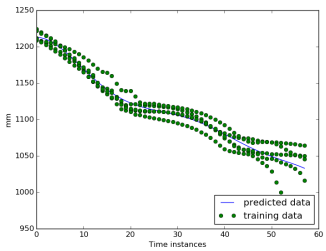

Figure A.1: Reconstructed X-axis using identity the matrix generalized with GP (letter A)

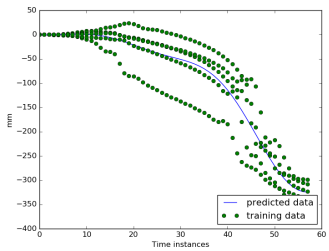

Figure A.2: Reconstructed Y-axis using identity the matrix generalized with GP (letter A)

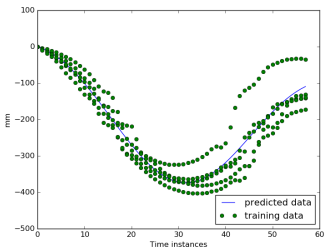

Figure A.3: Reconstructed Z-axis using identity the matrix generalized with GP (letter A)

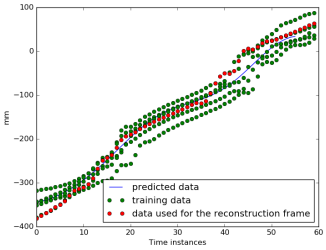

Figure A.4: Reconstructed X-axis using their respective initial frame generalized with GP (letter A)

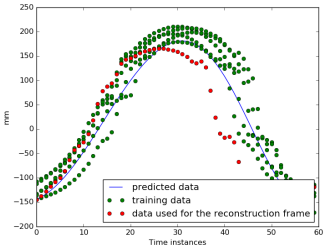

Figure A.5: Reconstructed Y-axis using their respective initial frame generalized with GP (letter A)

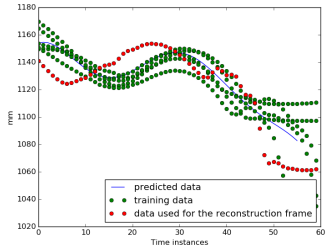

Figure A.6: Reconstructed Z-axis using their respective initial frame generalized with GP (letter A)

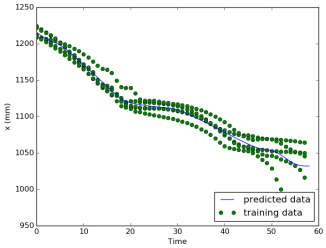

Figure A.7: Reconstructed X-axis using identity the matrix generalized with GMM (letter A)

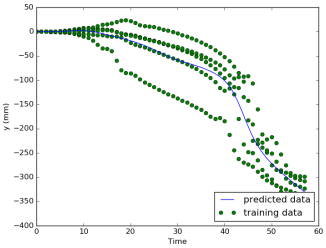

Figure A.8: Reconstructed Y-axis using identity the matrix generalized with GMM (letter A)

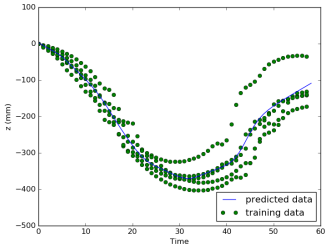

Figure A.9: Reconstructed Z-axis using identity the matrix generalized with GMM (letter A)

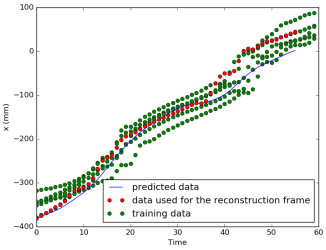

Figure A.10: Reconstructed X-axis using their respective initial frame generalized with GMM (letter A)

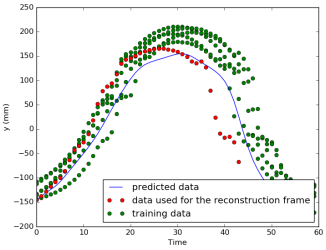

Figure A.11: Reconstructed Y-axis using their respective initial frame generalized with GMM (letter A)

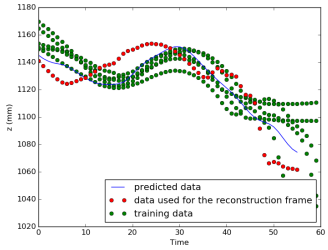

Figure A.12: Reconstructed Z-axis using their respective initial frame generalized with GMM (letter A)

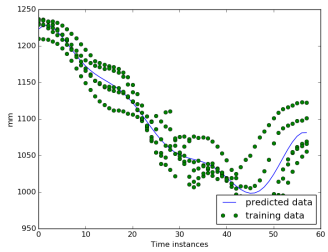

Figure A.13: Reconstructed X-axis using identity the matrix generalized with GP (letter N)

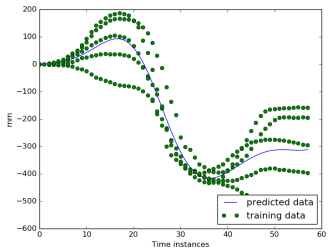

Figure A.14: Reconstructed Y-axis using identity the matrix generalized with GP (letter N)

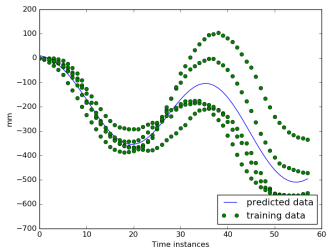

Figure A.15: Reconstructed Z-axis using identity the matrix generalized with GP (letter N)

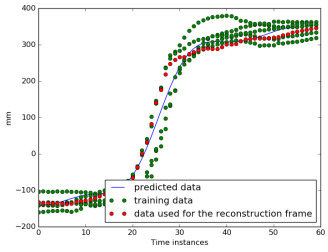

Figure A.16: Reconstructed X-axis using their respective initial frame generalized with GP (letter N)

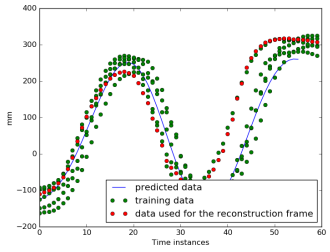

Figure A.17: Reconstructed Y-axis using their respective initial frame generalized with GP (letter N)

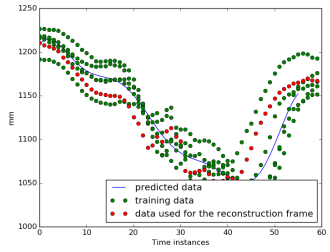

Figure A.18: Reconstructed Z-axis using their respective initial frame generalized with GP (letter N)

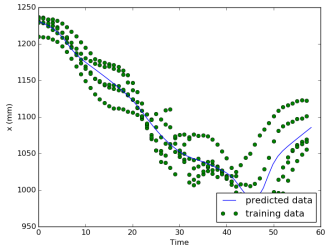

Figure A.19: Reconstructed X-axis using identity the matrix generalized with GMM
(letter N)

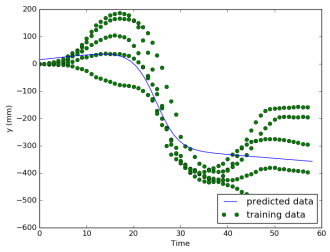

Figure A.20: Reconstructed Y-axis using identity the matrix generalized with GMM (letter N)

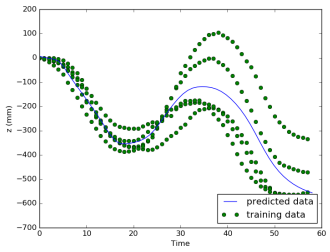

Figure A.21: Reconstructed Z-axis using identity the matrix generalized with GMM (letter N)

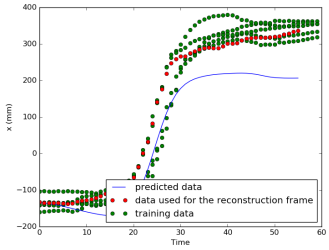

Figure A.22: Reconstructed X-axis using their respective initial frame generalized with GMM (letter N)

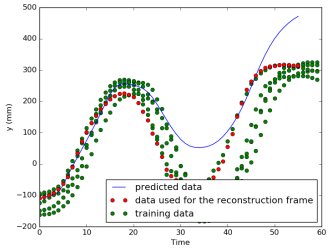

Figure A.23: Reconstructed Y-axis using their respective initial frame generalized with GMM (letter N)

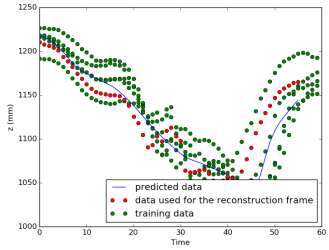

Figure A.24: Reconstructed Z-axis using their respective initial frame generalized with GMM (letter N)

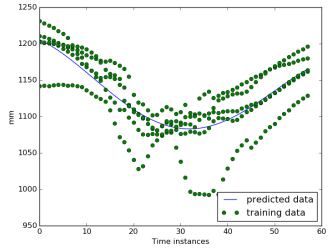

Figure A.25: Reconstructed X-axis using identity the matrix generalized with GP (letter O)

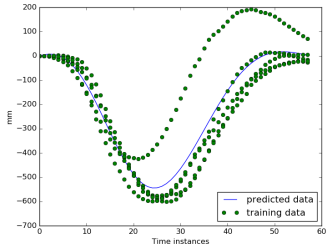

Figure A.26: Reconstructed Y-axis using identity the matrix generalized with GP (letter O)

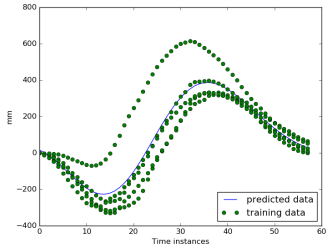

Figure A.27: Reconstructed Z-axis using identity the matrix generalized with GP (letter O)

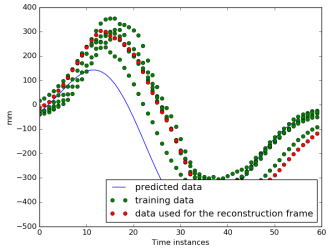

Figure A.28: Reconstructed X-axis using their respective initial frame generalized with GP (letter O)

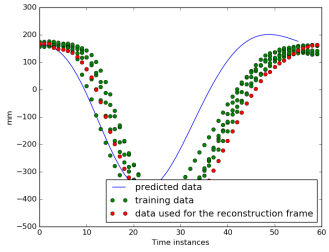

Figure A.29: Reconstructed Y-axis using their respective initial frame generalized with GP (letter O)

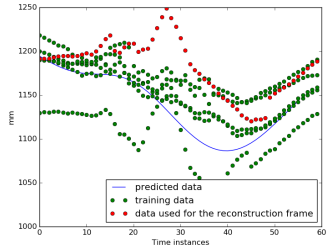

Figure A.30: Reconstructed Z-axis using their respective initial frame generalized with GP (letter O)

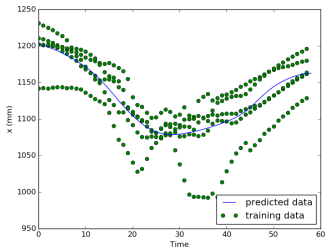

Figure A.31: Reconstructed X-axis using identity the matrix generalized with GMM (letter O)

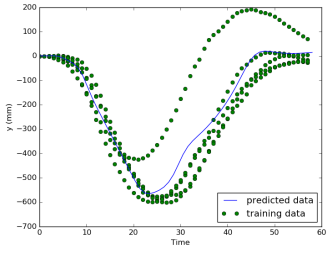

Figure A.32: Reconstructed Y-axis using identity the matrix generalized with GMM (letter O)

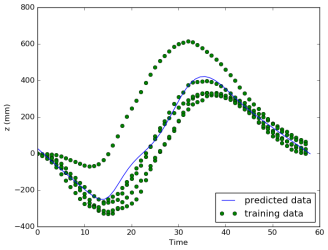

Figure A.33: Reconstructed Z-axis using identity the matrix generalized with GMM (letter O)

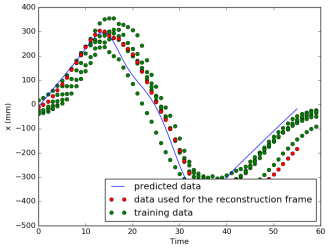

Figure A.34: Reconstructed X-axis using their respective initial frame generalized with GMM (letter O)

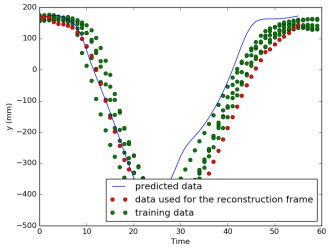

Figure A.35: Reconstructed Y-axis using their respective initial frame generalized with GMM (letter O)

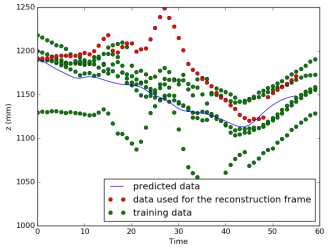

Figure A.36: Reconstructed Z-axis using their respective initial frame generalized with GMM (letter O)

A.2 Based on the position predict the velocity

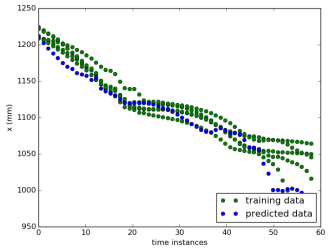

Figure A.37: Predict velocity for the X-axis based on a identity matrix reconstruction (letter A)

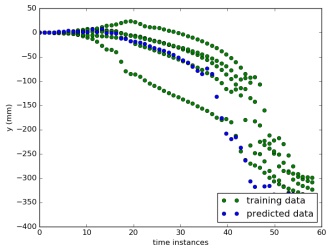

Figure A.38: Predict velocity for the Y-axis based on a identity matrix reconstruction (letter A)

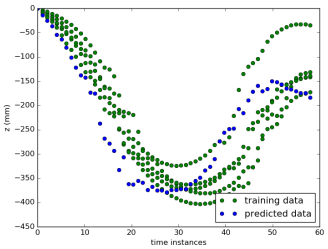

Figure A.39: Predict velocity for the Z-axis based on a identity matrix reconstruction (letter A)

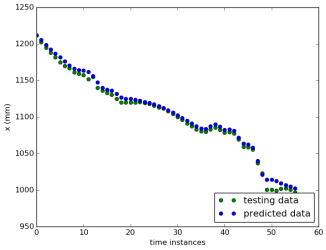

Figure A.40: Predict velocity for the X-axis based on the Cartesian reconstruction (letter A)

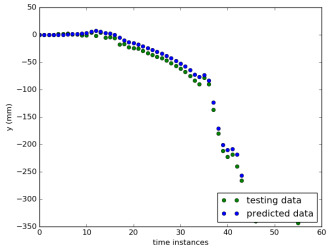

Figure A.41: Predict velocity for the Y-axis based on the Cartesian reconstruction (letter A)

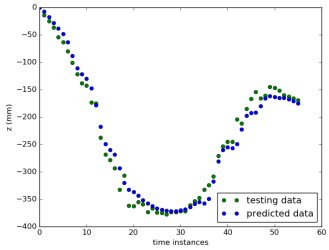

Figure A.42: Predict velocity for the Z-axis based on the Cartesian reconstruction (letter A)

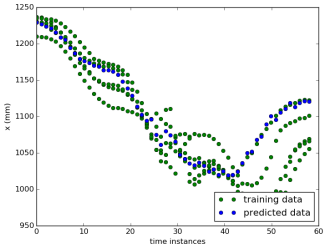

Figure A.43: Predict velocity for the X-axis based on a identity matrix reconstruction (letter N)

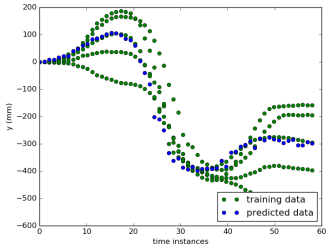

Figure A.44: Predict velocity for the Y-axis based on a identity matrix reconstruction (letter N)

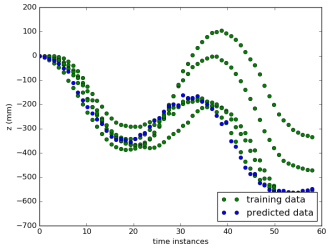

Figure A.45: Predict velocity for the Z-axis based on a identity matrix reconstruction (letter N)

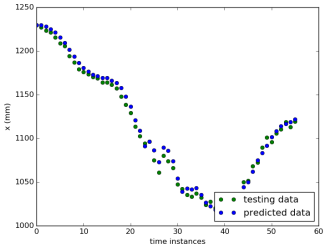

Figure A.46: Predict velocity for the X-axis based on the Cartesian reconstruction (letter N)

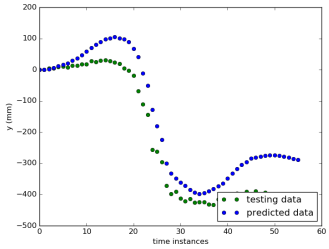

Figure A.47: Predict velocity for the Y-axis based on the Cartesian reconstruction (letter N)

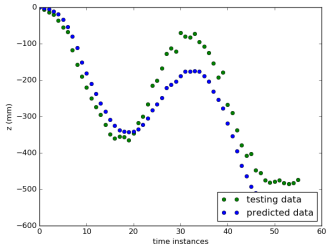

Figure A.48: Predict velocity for the Z-axis based on the Cartesian reconstruction (letter N)

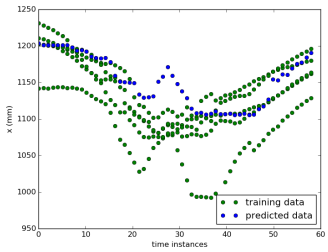

Figure A.49: Predict velocity for the X-axis based on a identity matrix reconstruction (letter O)

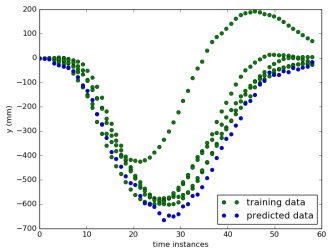

Figure A.50: Predict velocity for the Y-axis based on a identity matrix reconstruction (letter O)

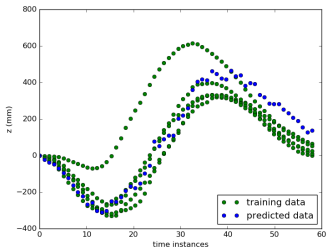

Figure A.51: Predict velocity for the Z-axis based on a identity matrix reconstruction (letter O)

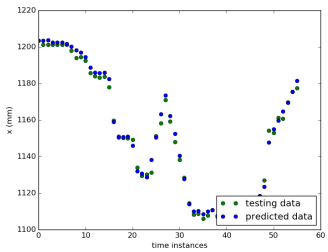

Figure A.52: Predict velocity for the X-axis based on the Cartesian reconstruction (letter O)

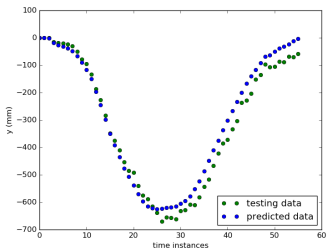

Figure A.53: Predict velocity for the Y-axis based on the Cartesian reconstruction (letter O)

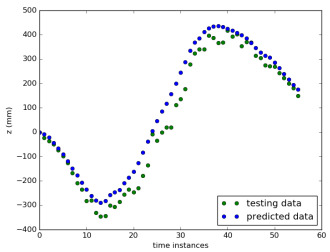

Figure A.54: Predict velocity for the Z-axis based on the Cartesian reconstruction (letter O)

List of Figures

Bibliography

[B+98] Jeff A Bilmes et al. A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. *International Computer Science Institute*, 4(510):126, 1998.

[BCDS08] Aude Billard, Sylvain Calinon, Ruediger Dillmann, and Stefan Schaal. Robot programming by demonstration. In *Springer handbook of robotics*, pages 1371–1394. Springer, 2008.

[Bor04] Sean Borman. The expectation maximization algorithm-a short tutorial. *Submitted for publication*, pages 1–9, 2004.

[BSP+04] Jernej Barbič, Alla Safonova, Jia-Yu Pan, Christos Faloutsos, Jessica K Hodgins, and Nancy S Pollard. Segmenting motion capture data into distinct behaviors. In *Proceedings of Graphics Interface 2004*, pages 185–194. Canadian Human-Computer Communications Society, 2004.

[CB08] Sylvain Calinon and Aude Billard. A probabilistic programming by demonstration framework handling constraints in joint space and task space. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 367–372. IEEE, 2008.

[CGB07] Sylvain Calinon, Florent Guenter, and Aude Billard. On learning, representing, and generalizing a task in a humanoid robot. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37(2):286–298, 2007.

[DB08] Chuong B Do and Serafim Batzoglou. What is the expectation maximization algorithm? *Nature biotechnology*, 26(8):897–899, 2008.

[DFR15] Marc Peter Deisenroth, Dieter Fox, and Carl Edward Rasmussen. Gaussian processes for data-efficient learning in robotics and control. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 37(2):408–423, 2015.

[Do08] Chuong B Do. The multivariate gaussian distribution, 2008.

[DS10] Joris De Schutter. Invariant description of rigid body motion trajectories. *Journal of Mechanisms and Robotics*, 2(1):011004, 2010.

[Ebd08] Mark Ebden. Gaussian processes for regression: A quick introduction. *The Website of Robotics Research Group in Department on Engineering Science, University of Oxford*, 2008.

[FRMS12] Chris Fraley, Adrian E Raftery, T Brendan Murphy, and Luca Scrucca. mclust version 4 for r: normal mixture modeling for model-based clustering, classification. Technical report, and Density Estimation Technical Report No, 2012.

[HGCB08] Micha Hersch, Florent Guenter, Sylvain Calinon, and Aude Billard. Dynamical system modulation for robot learning via kinesthetic demonstrations. *Robotics, IEEE Transactions on*, 24(6):1463–1467, 2008.

[KF09] Jonathan Ko and Dieter Fox. Gp-bayesfilters: Bayesian filtering using gaussian process prediction and observation models. *Autonomous Robots*, 27(1):75–90, 2009.

[KF11] Jonathan Ko and Dieter Fox. Learning gp-bayesfilters via gaussian process latent variable models. *Autonomous Robots*, 30(1):3–23, 2011.

[KPPB07] Kristian Kersting, Christian Plagemann, Patrick Pfaff, and Wolfram Burgard. Most likely heteroscedastic gaussian process regression. In *Proceedings of the 24th international conference on Machine learning*, pages 393–400. ACM, 2007.

[NH98] Radford M Neal and Geoffrey E Hinton. A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, pages 355–368. Springer, 1998.

[Ras06] Carl Edward Rasmussen. Gaussian processes for machine learning. 2006.

[Rey09] Douglas Reynolds. Gaussian mixture models. In *Encyclopedia of Biometrics*, pages 659–663. Springer, 2009.

[Rie86] JH Rieger. Three-dimensional motion from fixed points of a deforming profile curve. *Optics Letters*, 11(3):123–125, 1986.

[Sch06] Stefan Schaal. Dynamic movement primitives-a framework for motor control in humans and humanoid robotics. In *Adaptive Motion of Animals and Machines*, pages 261–280. Springer, 2006.

[SE10] Markus Schneider and Wolfgang Ertel. Robot learning by demonstration with local gaussian process regression. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 255–260. IEEE, 2010.

[SG00] Chris Stauffer and W. Eric L. Grimson. Learning patterns of activity using real-time tracking. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):747–757, 2000.

[SL13] Matteo Saveriano and Dongheui Lee. Invariant representation for user independent motion recognition. In *RO-MAN, 2013 IEEE*, pages 650–655. IEEE, 2013.

[SSL15] Raffaele Soloperto, Matteo Saveriano, and Dongheui Lee. A bidirectional invariant representation of motion for gesture recognition and reproduction. In *International Conference on Robotics and Automation*, 2015.

[VVK03] Jakob J Verbeek, Nikos Vlassis, and B Kröse. Efficient greedy learning of gaussian mixture models. *Neural computation*, 15(2):469–485, 2003.

License