Dynamic Movement Primitives in Latent Space of Time-Dependent Variational Autoencoders

Nutan Chen, Maximilian Karl, Patrick van der Smagt

***Abstract*— Dynamic movement primitives (DMPs) are powerful for the generalization of movements from demonstration. However, high dimensional movements, as they are found in robotics, make finding efficient DMP representations difficult. Typically, they are either used in configuration or Cartesian space, but both approaches do not generalize well. Additionally, limiting DMPs to single demonstrations restricts their generalization capabilities.**

In this paper, we explore a method that embeds DMPs into the latent space of a time-dependent variational autoencoder framework. Our method enables the representation of high-dimensional movements in a low-dimensional latent space. Experimental results show that our framework has excellent generalization in the latent space, e.g., switching between movements or changing goals. Also, it generates optimal movements when reproducing the movements.

I. Introduction

The representation of movement via Dynamic Movement Primitives (DMP) is a promising approach and is widely used for learning by demonstration and reinforcement learning [1]. The generalization of movements by DMP is robust to adaptation to different speed, shift, stretch and goal position by changing DMP parameters. A probabilistic version, dubbed ProMP [2], has been proposed which is capable of representing movement variance.

High-dimensional movement data increases the difficulty of learning, inverse kinematics and redundant degrees of freedom representation. In earlier work Gaussian Process Latent Variable Models (GPLVM) [3], [4] and denoising autoencoder [5] have been investigated for movement representation in latent space using DMP. These approaches efficiently reduced the data representation dimensions and enabled new movements generated by simply changing parameters in the latent space. However, these methods are only used to represent one, or at most two, different movement types; for more different movements, different models have to be trained. ProMP can combine and switch between different movements, and its dimension reduction model, dimension reduction ProMP (DR-ProMP) [6], was investigated for high-dimensional movements based on Expectation Maximization (EM). However, ProMP-based methods blend the movement via points which the trajectories go through, and require multiple demonstrations for each movement type to construct a sufficient probabilistic model.

In previous work [7] we have shown that, in a reinforcement learning setting, variational autoencoders (VAE) can build more meaningful latent spaces than autoencoders or PCA. Following that line of thought, here we propose to use a VAE rolled out in time to reduce the dimensions for DMPs. To this end, we use a technique called *Deep Variational Bayes Filtering* (DVBF) [8]. Our approach is an incremental advancement of DVBF and DMP by exploring DVBF to learn movements from multi-demonstrations, as well as benefiting from the latent representation of time-dependent VAE. Integration of DMPs with DVBF increases the constraints of the latent space, and therefore forces the distribution of the movements in the latent space to be meaningful. In our approach we train a model with a shared latent space for various similar as well as different movements. In contrast to ProMP, our method switches the movement completely, and does so by representing different movements in different “parts” of the latent space.

II. Method

Our approach consists of a modified DVBF which represents the high-dimensional data in a latent space and DMPs which learns movements from demonstrations in the latent space.

A. Dynamic Movement Primitives

DMPs are generally trained from a demonstration of a movement in its joint space or Cartesian space, which can then be re-used and adapted to the environment [1]. A DMP is a point attractor system written as a second-order dynamic model,

$$\tau\ddot{\mathbf{y}} = \alpha\big(\beta(\mathbf{g}-\mathbf{y}) - \dot{\mathbf{y}}\big) + \mathbf{f}, \tag{1}$$

where τ is a time constant and $\alpha > 0$ and $\beta > 0$ are damping constants. The trajectory $\mathbf{y}$ is attracted to the goal position $\mathbf{g}$ by the difference term $(\mathbf{g}-\mathbf{y})$. Usually, the last frame of the demonstration is set as the goal during training.

The forcing term $\mathbf{f}$ encodes the trajectory dynamics, which drives the system to the goal. Following [1], a basic version of $\mathbf{f}$ is chosen as a linear combination of basis functions Ψ_i,

$$\mathbf{f}_t = \frac{\sum_{i=1}^{N}\Psi_i(t)\mathbf{w}_i}{\sum_{i=1}^{N}\Psi_i(t)}, \tag{2}$$

where t represents the time steps. We focus on the discrete case, but an extension to rhythmic dynamical systems is straightforward by modifying the basis functions Ψ. We write a discrete Ψ as [1]

$$\Psi_i(t) = \exp\Big[-\frac{1}{2\sigma_i^2}(t-c_i)^2\Big], \tag{3}$$

[1]These authors are with the Faculty for Informatics, Technische Universität München, 80333 Germany `nutan.chen(at)gmail.com`, `karlma(at)in.tum.de`. PvdS is also with fortiss.

where constants c_i and σ_i are the width and centers of the Gaussian basis functions, respectively.

During training, with the demonstration $\mathbf{y}^{\text{demo}}$ and its computed derivatives, the target values of the forcing term results in,

$$\mathbf{f}^{\text{target}} = \tau^2 \ddot{\mathbf{y}}^{\text{demo}} - \alpha_z\big(\beta_z(\mathbf{g} - \mathbf{y}^{\text{demo}}) - \tau\dot{\mathbf{y}}^{\text{demo}}\big). \quad (4)$$

With $\{(\mathbf{f}_1, \mathbf{f}_1^{\text{target}}), \ldots, (\mathbf{f}_n, \mathbf{f}_n^{\text{target}})\}$ of length n, an optimizer can be used to minimize the discrepancy between $\mathbf{f}$ and $\mathbf{f}^{\text{target}}$

$$\mathbf{w}^\star = \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{i=1}^{n} L\big[\mathbf{f}_i, \mathbf{f}_i^{\text{target}}\big], \quad (5)$$

where L is a loss function.

B. Variational Autoencoder

In this section, we give a brief overview of variational autoencoders (VAE), which is the basic framework of time-dependent VAE for DMP.

1) Variational inference: Variational inference [9] is a method to approximate the intractable posterior distribution $p(\mathbf{z}|\mathbf{x})$ through a tractable approximate variational distribution $q_\phi(\mathbf{z})$. $\mathbf{x} \in \mathbb{R}^d$ and $\mathbf{z} \in \mathbb{R}^{d'}$ are the observed data and its corresponding latent representation, respectively. As the dissimilarity function, the Kullback-Leibler (KL) divergence between the approximate distribution q_ϕ and the target distribution p is minimized to obtain the variational parameter ϕ for the optimal approximate distribution q_ϕ. The marginal log-likelihood is written as

$$\log p(\mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z})}\left[\log \frac{p(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z})}\right] + \mathbb{KL}\big(q_\phi(\mathbf{z}) \parallel p(\mathbf{z}|\mathbf{x})\big). \quad (6)$$

$\log p(\mathbf{x})$ does not depend on q_ϕ and the KL term is non-negative; therefore, the first term, the variational lower bound $\mathcal{L}^{\text{bound}}(q)$ is maximized to minimize the KL divergence.

2) Variational autoencoder: A Variational autoencoder (VAE) [10] is a type of autoencoder [11] based on variational inference. The posterior distribution is defined as $p_\theta(\mathbf{z}|\mathbf{x}) \propto p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})$. The prior $p(\mathbf{z})$ is defined as an isotropic Gaussian distribution $\mathcal{N}(0, \mathbb{I}_D)$. The observation model $p_\theta(\mathbf{x}|\mathbf{z})$ is defined by a parameterized distribution, which is a generative neural network taking $\mathbf{z}$ as an input and outputs the parameters of the distribution. This distribution is chosen Gaussian for continuous data, or Bernoulli for binary data. θ are the weights of the neural network.

The approximate distribution $q_\phi(\mathbf{z}|\mathbf{x})$ is also a neural network which takes as input $\mathbf{x}$, outputs the parameters of the distribution over $\mathbf{z}$ and has the weights ϕ. It is called inference network or recognition network.

ϕ and θ can be jointly updated by optimization through back-propagation. VAE maximizes the variational lower bound to optimize the parameters θ and ϕ.

$$\begin{aligned}\log p_\theta(\mathbf{x}) &\geq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\right] \quad (7)\\ &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - \mathbb{KL}\big(q_\phi(\mathbf{z}|\mathbf{x}) \parallel p(\mathbf{z})\big),\end{aligned}$$

where the first term can be considered as a reconstruction loss, and the second is a regularization term which constrains the approximate posterior $q_\phi(\mathbf{z}|\mathbf{x})$ to be close to the prior $p(\mathbf{z})$.

C. DMP in latent space

The VAE, as formulized in the previous section, has no internal state, and therefore cannot represent temporal dependencies in the input data. There are several ways of dealing with this; one possibility is extending the VAE to be a recurrent neural network [12], [13]. While having their merit in, e.g., anomaly detection [14], their prediction capabilities are not as good as expected [15]. Furthermore, it is not clear how a control signal can be included.

A different, very promising approach is obtained by rolling a VAE out in time. This approach, called *Deep Variational Bayes Filtering* (DVBF) was published in [8]. In this paper we use DVBF to embed DMPs in VAE (see Fig. 1). By using this approach, we can directly learn all parameters—including the DMP parameters—using back-propagation.

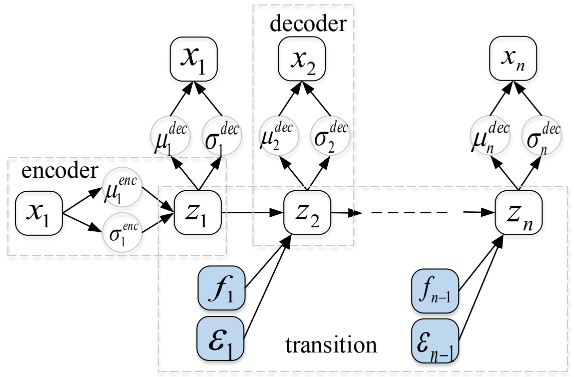


Fig. 1: VAE-DMP information flow for the generative movement.

1) Problem Formulation: Given a movement $\mathbf{x}_{1:n} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$, where n is the length of the movement, we want to model the movement in latent space $\mathbf{z}_{1:n} = \{\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_n\}$. In order to embed DMP into DVBF, (1) is formulated for the transition function from $\mathbf{z}_t$ to $\mathbf{z}_{t+1}$ in the latent space as

$$\begin{aligned}\tau\ddot{\mathbf{z}}_{t+1} &= \alpha\big(\beta(\mathbf{z}^{\text{goal}} - \mathbf{z}_t) - \dot{\mathbf{z}}_t\big) + \mathbf{f}_t + \epsilon\\ \dot{\mathbf{z}}_{t+1} &= \ddot{\mathbf{z}}_{t+1}\mathrm{d}t + \dot{\mathbf{z}}_t\\ \mathbf{z}_{t+1} &= \dot{\mathbf{z}}_{t+1}\mathrm{d}t + \mathbf{z}_t \end{aligned} \quad (8)$$

where $\mathrm{d}t$ is the step duration, $\mathbf{z}_t$ is the movement in latent space at step t, $\mathbf{z}^{\text{goal}}$ is the goal in the latent space corresponding to the final frame of movement in the joint space $\mathbf{x}_n$, $\mathbf{f}_t \in \mathbb{R}^{d'}$ is modified from (2) by changing the dimension to fit the latent space, $\epsilon = \hat{\epsilon}\mathbf{w}_\epsilon$, $\hat{\epsilon} \sim \mathcal{N}(0, \Sigma_\epsilon)$ is the system noise, and $\mathbf{w}_\epsilon \in \mathbb{R}^{d' \times d'}$ is the weight for the noise, so that the scale of the noise is learned autonomously.

Eq. (8) can be simplified as

$$\begin{bmatrix}\mathbf{z}_{t+1}\\ \dot{\mathbf{z}}_{t+1}\end{bmatrix} = \mathbf{A}\begin{bmatrix}\mathbf{z}_t\\ \dot{\mathbf{z}}_t\end{bmatrix} + \mathbf{b}, \quad (9)$$

where the transition matrix is computed as

$$\mathbf{A} = \begin{bmatrix} -\mathrm{dt}\alpha\beta\frac{1}{\tau} & -\mathrm{dt}\alpha\frac{1}{\tau}+1 \\ -\alpha\beta\frac{1}{\tau} & -\alpha\frac{1}{\tau} \end{bmatrix} \mathrm{dt} + I, \tag{10}$$

and the control input is defined as

$$\mathbf{b} = \begin{bmatrix} \mathrm{dt} \\ 1 \end{bmatrix} (\alpha\beta\mathbf{z}^{\mathrm{goal}} + \mathbf{f}_t + \hat{\epsilon})\mathrm{dt}\frac{1}{\tau}. \tag{11}$$

I is an identity matrix.

Instead of generating $\mathbf{f}^{\mathrm{target}}$ as (4) and (5) to find its parameters, we embed the DMP transition into DVBF and train its parameters through backpropagation.

2) Inference model: The encoder network is only used for the initial and goal states. We take a diagonal Gaussian distribution with mean μ_t and covariance $\mathbf{\Sigma}_t$. The encoding process is $q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}|\mu_t, \mathrm{diag}(\sigma_t^2))$.

The mean and covariance are encoded by a neural network

$$\mu_t^{\mathrm{enc}} = \mathbf{w}_\mu^{\mathrm{enc}} h_\phi(\mathbf{x}_t) + \mathbf{b}_\mu^{\mathrm{enc}},$$
$$\left(\sigma_t^{\mathrm{enc}}\right)^2 = \left(\mathbf{w}_\sigma^{\mathrm{enc}} h_\phi(\mathbf{x}_t) + \mathbf{b}_\sigma^{\mathrm{enc}}\right)^2, \tag{12}$$

where, h_ϕ denotes an activation function. $\mathbf{w}_\mu^{\mathrm{enc}}$, $\mathbf{w}_\sigma^{\mathrm{enc}}$, $\mathbf{b}_\mu^{\mathrm{enc}}$ and $\mathbf{b}_\sigma^{\mathrm{enc}}$ are parameters. Based on a deep neural network, we parameterize the mean and variance of a Gaussian distribution.

We can get another representation of a latent space of the initial and goal frames by a transition function $\mathbf{z}_t^\star = g(\mathbf{z}_t)$. The transition function g can be, e.g., a multilayer perceptron (MLP). In the sequel, we will use $\mathbf{z}^\star$ in lieu of $\mathbf{z}$. The transition forces the Gaussian-shaped latent space to be replaced by any other kind of shapes.

During training, ϵ_{t+1} is predicted from $\mathbf{z}_t$ and $\mathbf{x}_{t+1}$, while ϵ is sampled from the prior without $\mathbf{z}_t$ or $\mathbf{x}_{t+1}$ for generating movement after training. ϵ is diagonal Gaussian distribution

$$\epsilon_{t+1} \sim p_\gamma(\epsilon_{t+1}|\mathbf{x}_{t+1}, \mathbf{z}_t) = \mathcal{N}(\mu_t^{\mathrm{noise}}, \Sigma_t^{\mathrm{noise}}), \tag{13}$$

where γ are the weights of neural network. The mean μ_t^{noise} and covariance $\Sigma_t^{\mathrm{noise}}$ are encoded by a neural network.

3) Generative model: We segment a movement into subsequence with length of l. The generative model includes decoders with $l = 1$

$$q_\phi(\mathbf{z}_t|\mathbf{x}_t) = \mathcal{N}(\mu_t^{\mathrm{enc}}, \Sigma_t^{\mathrm{enc}}), \tag{14}$$

and with $l > 1$

$$\hat{q}_{\hat{\phi}}(\mathbf{z}_{t+1}|\mathbf{z}_t, \mathbf{f}_t, \epsilon_t). \tag{15}$$

Consequently, the generated observation $\hat{\mathbf{x}}$ is

$$\hat{\mathbf{x}}_t \sim p_\theta(\mathbf{x}_t|\mathbf{z}_t) = \mathcal{N}(\mu_t^{\mathrm{dec}}, \Sigma_t^{\mathrm{dec}}). \tag{16}$$

(15) is the transition from $\mathbf{z}_t$ to $\mathbf{z}_{t+1}$, which is described in the next subsection.

We take a Gaussian distribution with mean and constant covariance. Reconstruction ("decoding") of $\mathbf{x}$ is by another neural network,

$$\mu_t^{\mathrm{dec}} = \mathbf{w}_\mu^{\mathrm{dec}} h_\theta(\mathbf{z}_t) + \mathbf{b}_\mu^{\mathrm{dec}},$$
$${\sigma_t^{\mathrm{dec}}}^2 = {\mathbf{b}_\sigma^{\mathrm{dec}}}^2 \tag{17}$$

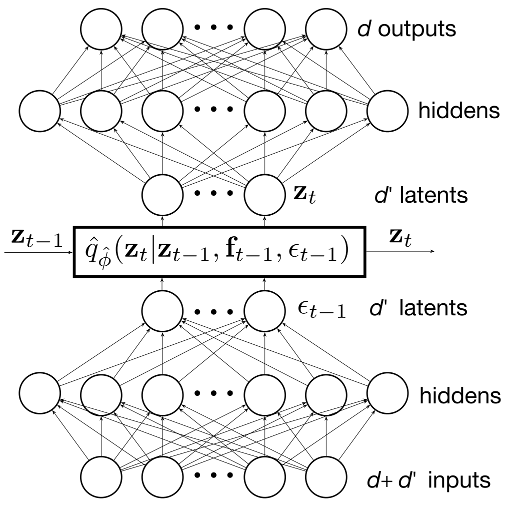


Fig. 2: Neural network structure of the time-dependent VAE in time step t. The network structure is similar to a regular autoencoder with the addition of the DMP transition function in the center. The nodes ϵ_{t-1} are stochastic with mean and variance created by the neural network. The autoencoder takes $\mathbf{z}_{t-1}$ together with $\mathbf{x}_t$ as input.

where h_θ is the activation function. $\mathbf{w}_\mu^{\mathrm{dec}}$, $\mathbf{w}_\sigma^{\mathrm{dec}}$, $\mathbf{b}_\mu^{\mathrm{dec}}$ and $\mathbf{b}_\sigma^{\mathrm{dec}}$ are parameters.

Reconstruction through $\mathbf{z}$ from both (14) and (15) enforces the $\mathbf{z}_{t+1}$ from the latent transition to approximate the $\mathbf{z}_{t+1}$ from the encoder. Therefore, the training model contains both $\mathbf{z}$'s of subsequence length of 1, which allows $\mathbf{x}_t$ of every time step to be reconstructed from $\mathbf{z}_t$ of (14), and subsequence length of n.

4) Transition model: In the latent space, we predict the local transformation parameters of $\mathbf{f}$ and ϵ (see Fig. 2). The transition is described as

$$\mathbf{z}_{t+1} = f(\mathbf{z}_t, \mathbf{f}_t, \epsilon_t), \tag{18}$$

where f is a function of (9). ϵ is sample-specific noise. With meaningful transition priors, it avoids overfitting and has meaningful manifolds in the latent space.

The initial $\dot{\mathbf{z}}$ of the starting of the sequence can be predicted directly from the encoder with $\mathbf{x}_{1:m}$ as the input and $\{\mathbf{z}, \dot{\mathbf{z}}\}$ as the output, where $1 < m \leq n$. Besides, another approach initializes $\mathbf{z}_0$ through the first step $\mathbf{z}_1$ and its previous step $\mathbf{z}_0$ using (8).

D. Learning

The learning process is through stochastic gradient variational Bayes.

Based on DVBF, the lower bound is rewritten as,

$$\mathcal{L}(\mathbf{x}_{1:n}, \theta, \phi|\mathbf{f}_{1:n})$$
$$= \mathbb{E}_{q_\phi}\left[\log p_\theta(\mathbf{x}_{1:n}|\mathbf{z}_{1:n}) - \log q_\phi(\epsilon_{1:n}|\mathbf{x}_{1:n}) + \log p(\epsilon_{1:n})\right] \tag{19}$$
$$= \mathbb{E}_{q_\phi}\left[\log p_\theta(\mathbf{x}_{1:n}|\mathbf{z}_{1:n})\right] - \mathrm{KL}(q_\phi(\epsilon_{1:n}|\mathbf{x}_{1:n})\|p(\epsilon_{1:n})), \tag{20}$$

where $\epsilon_{1:n} = \{\epsilon_1, \epsilon_2, ..., \epsilon_n\}$ and $\mathbf{f}_{1:n} = \{\mathbf{f}_1, \mathbf{f}_2, ..., \mathbf{f}_n\}$. $\mathbf{f}$ is encoded into $\mathbf{z}$ in (19).

During the training process, ϵ can act as a shortcut to encode all of the dynamical information; as a result, $\mathbf{f}$ is not learning meaningful values. The annealing schedule improves the training of $\mathbf{f}$ and smooths out these local minima [16]. (19) is written as

$$\begin{aligned}\mathcal{L}(\mathbf{x}_{1:n}, \theta, \phi|\mathbf{f}_{1:n}) =& \mathbb{E}_{\mathbf{q}_\phi}\big[c_{t_a} \log p_\theta(\mathbf{x}_{1:n}|\mathbf{z}_{1:n}) \\ & - \log q_\phi(\epsilon_{1:n}|\mathbf{x}_{1:n}) + c_{t_a} \log p(\epsilon_{1:n})\big],\end{aligned} \tag{21}$$

where $c_{t_a} = \max(1, 0.01 + t_a/T_a)$ and t_a increases linearly from 0 after every training epoch until c_{t_a} equals 1.

Having only a small number of demonstrations increases the difficulty of training, since Monte Carlo estimation is used in (18) as in [10]. Additionally, a long sequence demonstration increases the training time and weakens the structure of the latent space at the start of the sequence, because of vanishing and exploding gradients. Therefore, we split a sequence up into overlapping subsequences to increase the batch size and shorten the sequence. On the contrary, the model has difficulties in recognizing the dynamics of the movement with too short subsequences, and consequently, the latent space might be unstructured. Thus, as a hyperparameter, the length of the subsequences, l_{sub}, can be searched during training.

E. Multi-demonstration model

We extend VAE-DMP by training a single, shared latent space with multiple motion sequences. With multiple motion sequences [17], the model (a) is not limited by the number of sequences, and (b) is more adaptable to a new movements, e.g., movement switching or goal changing.

Instead of learning the weights $\mathbf{w}$ for each type of movement individually we use a neural network which recognizes the correct weights from the observations. That way we get a continuous set of weights where the mapping between movement type and weights is trained by backpropagation. This neural network has the following shape:

$$\mathbf{w}(\mathbf{x}_{1:n}) = g_2\big(g_1(\mathbf{x}_1), g_1(\mathbf{x}_2), g_1(\mathbf{x}_3), \ldots, g_1(\mathbf{x}_n)\big) \tag{22}$$

where g_2 and g_1 are each fully connected neural networks. We will call this whole network MLP$_w$.

F. Movement switching

The first generalization of the model is switching the movements after training by simply switching the weight $\mathbf{w}$ of $\mathbf{f}$ and the goal $\mathbf{z}_g$. We have $\eta^i \in [0, 1]$ for movement i. The weight and goal of the new movements at t step are

$$\mathbf{w}_t^\star = \sum_i \left(\frac{\eta_t^i \mathbf{w}_t^i}{\sum_i \eta_t^i}\right), \quad \mathbf{z}_{gt}^\star = \sum_i \left(\frac{\eta_t^i \mathbf{z}_{gt}^i}{\sum_i \eta_t^i}\right). \tag{23}$$

Switching from movement A to movement B, η^A is changed from 1 to 0, and η^B from 0 to 1. Given the starting point and the duration of switching, η^A and η^B can be estimated.

Goal changing enables the latent value to transfer from movement A to movement B, while the force term enables the movement to follow the demonstration trajectory B after switching. If we only change the goal but not the force term, the movement also switches from A to B; however, the trajectory is quite random after switching.

G. Goal changing

The flexible properties of DMPs are retained in VAE-DMP. For instance, it is able to reach new set points by simply changing the goal after training, while keeping the invariant properties. Following [1], to keep the invariant properties, an extra term of $(\mathbf{g} - \mathbf{y_0})$ is multiplied to the right side of (2). In addition, it can be multiplied by $(\mathbf{g} - \mathbf{y_0})/(|\mathbf{g}_{\text{fit}} - \mathbf{y}_{0,\text{fit}}| + \eta)$ instead to avoid coinciding $\mathbf{y}_0$ and $\mathbf{g}$, where fit represents the training data and η is a very small positive constant.

Other reformulations of DMP, e.g. to do obstacle avoidance [18], can be implemented in VAE-DMP straightforwardly, but here we only focus on goal changing.

III. Experiments

Experiments with two data sets are performed to evaluate VAE-DMP. The first data set contains optical tracking data of human movements. The second data set is obtained from 6-DoF robot arm simulations.

A. High-dimensional human movement

The data set for the first set of experiments is the CMU Graphics Lab Motion Capture Database, which is a part of the *KIT Whole-Body Human Motion Database*[1]. These data are downsampled and preprocessed as described in [5], and the resulting data consist of multiple movements, each of which is coded in 70 time steps of 50-dimensional vectors. The data is normalized to zero mean for every joint. Since we focus on learning relative movements, the body center is fixed. We split the movements up into subsequences with $l_{sub} = 10$ time steps.

The DMPs are set to critical control by setting $\beta = \alpha/4$ [1]. The VAE architecture is 5 layers with d inputs, 200 hidden neurons, d' latents, 200 hidden neurons, d outputs, where rectifier and identity activations are used for the hidden layers and the output layer, respectively. The structure of the VAE is shown in Fig. 2. The neural network g_2 inside the MLP$_w$ network has $70 \times h$ inputs and $50 \times d'$ outputs. The g_1 network takes 50 inputs and outputs h. Where h is 10 for the humanoid experiments and 5 for the robot experiment. Both g_1 and g_2 have no hidden layers. g_1 uses softmax as activation function and g_2 uses the identity activation function. The structure and sizes of the MLP$_w$ network can be seen in Fig. 3. The hyperparameters of DMP and VAE are chosen based on the reconstruction error and the training time by grid search.

1) Learning different movements: In this task we train multiple demonstrations of walking, kicking, taichi and punching from 5 subjects (viz. subject 35, 74, 49, 120, and 143), all in one and the same model, and compare the results when we train each movement type to a single, specialised

[1]`https://motion-database.humanoids.kit.edu`

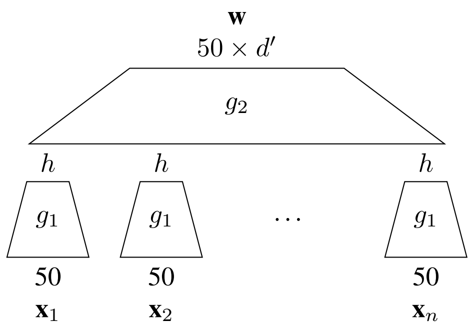


Fig. 3: Structure of MLP$_w$ with input and output sizes. The g_1 networks take the 50-dimensional input $\mathbf{x}_t$ and has an output with h dimensions. g_2 has $70 \times h$ inputs and $50 \times d'$ outputs. h is 10 for the humanoid experiments and 5 for the robot experiments. d' is the number of latent dimensions.

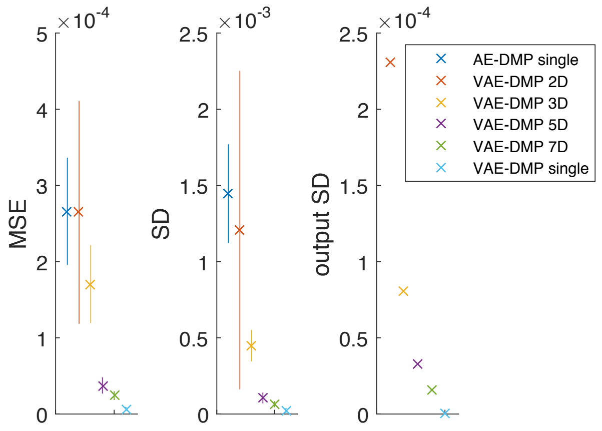


Fig. 4: Error of the reconstruction of the movement modeling. The error is MSE ± SD of every single subject averaged over all joints over a whole movement in radians. The result is evaluated using Mean square error (MSE ± standard deviation). The left and middle figures show MSE and SD averaged over 5 subjects. The right figure is the mean output SD (directly predicted from the encoder) of the VAE-DMP reconstruction. It is a constant value for every joint during the whole movement, so that output SD of VAE-DMP 2D, 3D, 5D, and 7D do not have variance.

model (called "VAE-DMP single"). We also compare the results to our previous "AE-DMP single" [5], in which we train one movement per autoencoder. Fig. 4 shows the pose reconstruction error of our previous work of AE-DMP single with 5D latent space, VAE-DMP with 2D, 3D, 5D and 7D latent space, and VAE-DMP single with 5D latent space. For VAE-DMP single we used 120 rather than 200 hidden units in each hidden layer. Noticeable is the improvement over our previous model, AE-DMP single. When sufficient latent variables are chosen (in this case, 5 or 7), the error is much lower, even for the approach where all poses are represented in one model. The lowest reconstruction error is obtained in VAE-DMP single, with 7D VAE-DMP a close second.

Although the reconstruction accuracy of VAE-DMP with

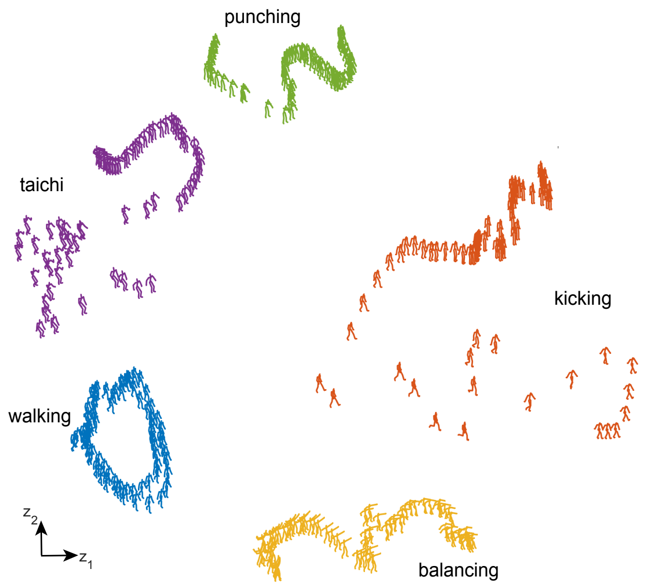


Fig. 5: Movement distribution in 2D latent space. z_1 and z_2 are two latent dimensions, and every body pose in the joint space is generated from its corresponding latent state.

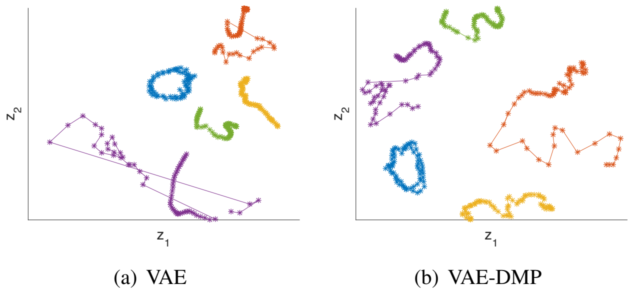


Fig. 6: Trajectories of five human movements in the 2D latent space using VAE and VAE-DMP. The movements are colored the same as Fig. 5

2D latent space yields to that with 7D latent space, we can more easily plot the former. Fig. 5 therefore illustrates the distribution of five movements in the latent space of VAE-DMP 2D. The patterns of the various generative movements can be seen. The walking is periodic, while punching is a single line in latent space. Kicking is a large-range movement, so that it has large range in the latent space, while balancing only has relative slight movement, and the range in the latent space is small.

The latent space of VAE-DMP is more meaningful than that of VAE (see Fig. 6). In the VAE latent space, a sequence of movement may spread far with different spacings for complex movements such as taichi, since it does not encode time information into the latent space. Accordingly, big gaps between two time steps may cause difficulties for DMPs. In addition, the a large geometry distance in joint space may result in a small distance in the VAE latent space such as kicking. With correctly encoding the geometry distance from the joint space, VAE-DMP can improve the multi-

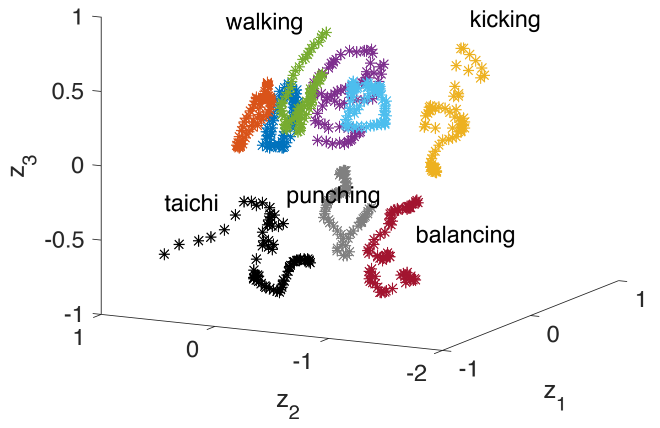


Fig. 7: Nine movements represented in the 3D latent space of a single VAE-DMP.

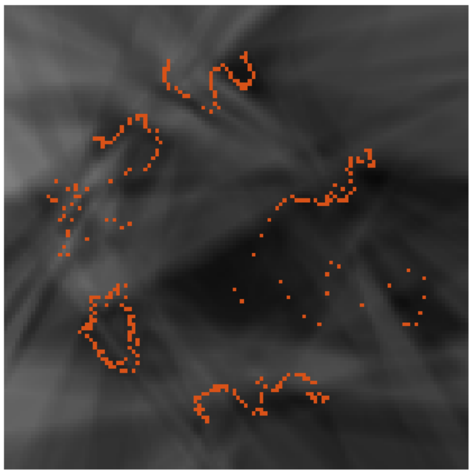


Fig. 8: Smoothness of the latent space. The plot represents values of dx/dz, which vary between 4.7 and 7.4 for a 2D VAE-DMP. Higher values—indicated by lighter areas—mean that a step in latent space corresponds to a larger body movement in x. The colored points correspond to the demonstrated movements, as in Fig. 5. The data are evaluated at 120×120 grid points in the 2D latent space.

demonstration model ability compared with VAE.

Training with additional 4 walking movements from other subjects (viz. Subject 86, 91, 114 and 139), we have the results shown in Fig. 7; in this case, a VAE-DMP 3D is used. The five walking movements distribute in a cluster, disjunct from the other movements. The leg movements cluster approximately on the positive of the z_3 axis, while the arm movements cluster on the negative of the z_3 axis.

2) Latent space smoothness: The latent space of a VAE-DMP codes the learned movements in a compact, multivariate Gaussian space. As seen in Fig. 5, the demonstration trajectories do not span the whole latent space. But what movements are coded between the demonstrated trajectories? To verify that no discontinuities or "large jumps" occur when sampling a trajectory in latent space, we evaluate dx/dz over the whole latent space of a VAE-DMP. The result for a

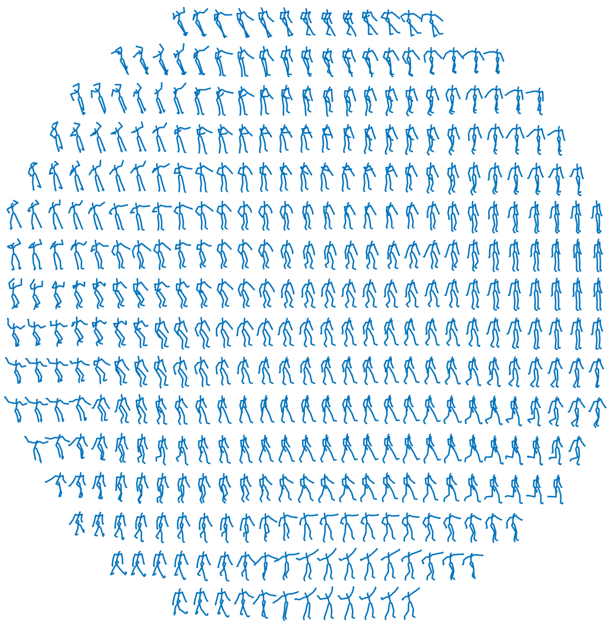


Fig. 9: Body postures in the 2D latent space, in the space spanned by the multivariate Gaussian. The area covering a variance of $\sigma = 1$ is plotted.

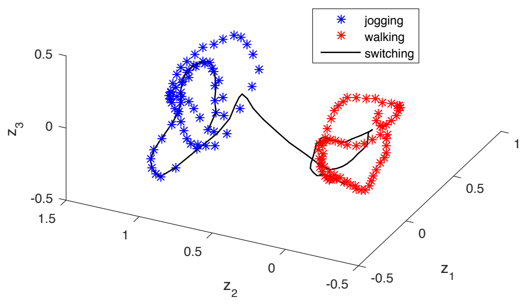
VAE-DMP 2D is shown in Fig. 8. In the whole latent space, dx/dz varies between 4.7 and 7.4 and is indeed very smooth. Similar smooth latent spaces are seen in VAE-DMP 3D and VAE-DMP 5D.

The joint space movements generated in 2D latent space is illustrated in Fig. 9. As $\mathbf{z}$ represents a multivariate normal $(\mu = 0, \sigma = 1)$ distribution, we can immediately compute the probability of a certain movement from its $\mathbf{z}$. As the plot shows, the movements between the demonstrated movements are smoothly interpolated. But also movements beyond that represent viable positions. Only when we move far away from the mean, starting between 2σ and 3σ does the latent space represent nonsensical postures. Of course, we can increase the training data set to obtain a larger confidence interval.

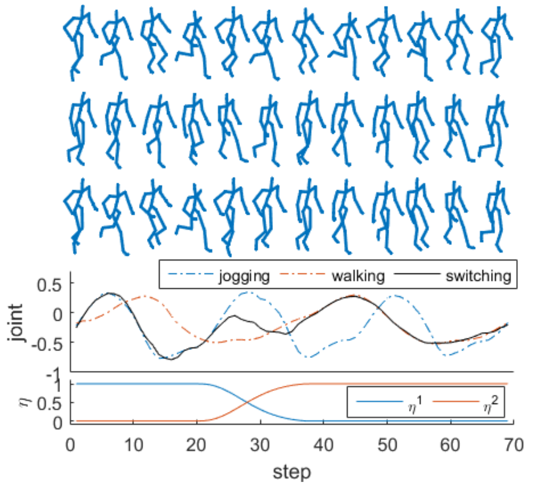
3) Movement switching: A model is trained by multiple demonstrations of a walking and jogging. $i = 1$ and $i = 2$ represent jogging and walking in η^i, respectively. The movements are switched from step 20 until step 38 (see Fig. 10). The starting and the duration of the switching can be changed. The model has a 5D latent space, and the largest two variance of the latent space are plotted. After it adapts to the walking from jogging, it follows the demonstration trajectory of walking. The generated latent values are not necessary to reproduce the latent values of demonstration precisely, while the reconstruction ability is more significant.

B. Robot simulation for goal changing

In this experiment we simulate a 6-DoF KUKA robot using the Robotic Toolbox [19]. In the data set, the length of a demonstration is 76 steps. The subsequence length is 10 time steps. For the representation of the movement in the



(a) Movement switching in latent space.



(b) Movement switching observed in state space. The figures of top three rows are plotted by every 6 time steps. We choose left hip for representing the joint, since it exactly follows the walking and jogging rhythm.

Fig. 10: Movement switching from walking to jogging.

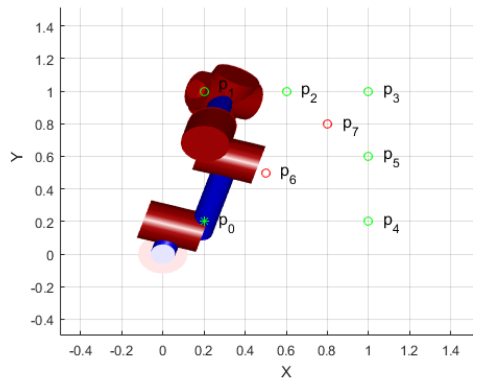


Fig. 11: Robot data set.

VAE, we use 5 layers with d, 100, $d' = 2$, 100, d. The input dimension of g_1 is d and the output dimension h is 5. g_2 takes $n \times h$ inputs and has $50 \times d'$ outputs.

The demonstration $\mathbf{x}^i$ is generated by moving the end-effector linearly from a starting point $\mathbf{p}_0$ to subsequent points $\mathbf{p}_i$, $i \in \{1, 2, \ldots, 5\}$ in Cartesian space. $\mathbf{x}^1$ is the demonstration for DMP, while $\{\mathbf{x}^1, \mathbf{x}^2, ..., \mathbf{x}^5\}$ are the demonstrations for VAE-DMP (see Fig. 11). $\mathbf{p}_6$ and $\mathbf{p}_7$ are the results of changing the goal to $[x = 0.5, y = 0.5]$ and $[x = 0.8, y = 0.8]$, respectively using VAE-DMP with given the force $\mathbf{f}^1$. All movements were planar, keeping z constant at 0.3.

For method comparison, we use the same version of DMP for both VAE-DMP and DMP. As described in Sec. II-G, to keep the invariant properties [1] for changing the goal, both forcing terms $\mathbf{f}$ in VAE-DMP and DMP are multiplied by a scaling term, specifically, $(\mathbf{z}^{\text{goal}} - \mathbf{z}_0)$ for VAE-DMP cq. $(\mathbf{g} - \mathbf{y_0})$ for DMP.

Fig. 12 shows the results of both DMP and VAE-DMP when changing the goal to [0.8, 0.8]. The optimal trajectory is the movement of the new goal with the end-effector moving linearly from $\mathbf{p}_0$. The optimal trajectory is not shown in the training data set but only its final frame is given as the new goal. It can be seen that VAE-DMP is able to generate a movement which is close to the optimal trajectory for the goal changing. However, DMP can only keep invariance of the demonstration for every joint independently, the effect of which is most clearly seen in joints 2 and 4. In this case, the robot joints are highly possible to be out of range. In contrast, VAE-DMP correlates the joints. In this experiment, VAE-DMP learns the invariance of the end-effector instead of every single joint. DMP requires manual selection of the important feature (e.g., joint angles of a robot or Cartesian space of the end effector) to learn, but VAE-DMP is able to learn the optimal trajectory autonomously and reproduces natural movements.

In joint 5, the start and end angle are almost the same in the demonstration. This makes generalization with DMP difficult, as the force term is almost zero. In VAE-DMP, the joints are correlated in the latent space, so that the beginning and ending values are not the same.

A video of the results is accompanied, which can be downloaded at: `https://brml.org/projects/dvbf`

IV. Conclusions

In this paper, we presented a novel approach to embed DMPs into a time-dependent variational autoencoder. Using a new approach called *Deep Variational Bayes Filtering*, we can embed DMPs in the latent space of a variational autoencoder, while simultaneously representing a large range of different movements in one single DVBF network. Thus we can also create smooth transitions between different movements. While DMPs can only independently keep the invariance per single joint, our approach allows the model to unsupervisedly learn the invariance features of the trajectory.

Our next step in this work is to validate this work with larger parts of the *KIT Whole-Body Human Motion Database*. Furthermore, we plan to investigate its use in humanoid motion planning.

V. Acknowledgements

This work has been supported in part by the TAC-MAN project, EC Grant agreement no. 610967, within the FP7 framework programme. The data used in this project can be found at `https://motion-database.`

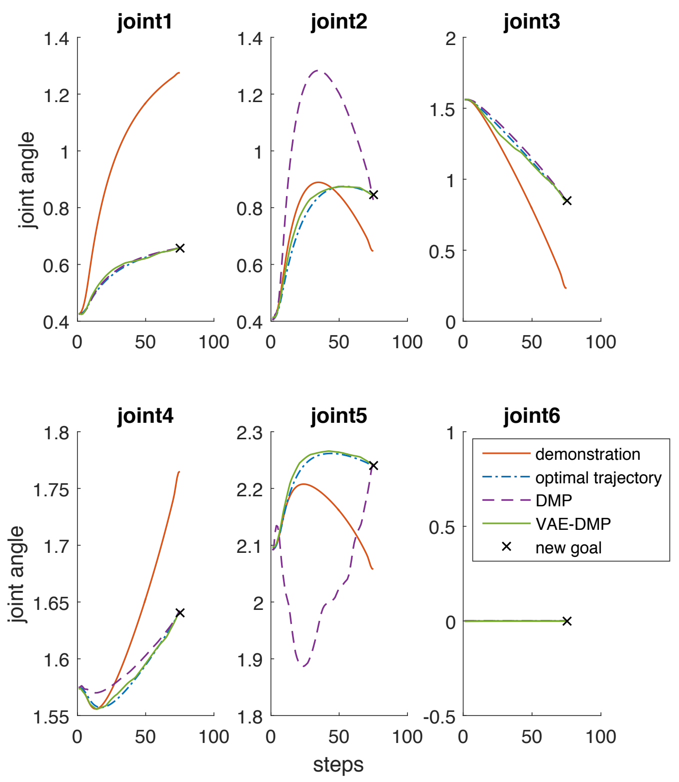


Fig. 12: Results of goal changing.

humanoids.kit.edu or at http://mocap.cs.cmu.edu. The database was created with funding from NSF EIA-0196217.

References

[1] A. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: learning attractor models for motor behaviors," *Neural computation*, vol. 25, no. 2, p. 328373, 2013.

[2] A. Paraschos, C. Daniel, J. Peters, and G. Neumann, "Probabilistic movement primitives," in *Advances in Neural Information Processing Systems*, 2013, pp. 2616–2624.

[3] N. D. Lawrence and A. J. Moore, "Hierarchical gaussian process latent variable models," in *Proceedings of the 24th international conference on Machine learning*. ACM, 2007, pp. 481–488.

[4] S. Bitzer and S. Vijayakumar, "Latent spaces for dynamic movement primitives," in *9th IEEE-RAS International Conference on Humanoid Robots*, Dec 2009, pp. 574–581.

[5] N. Chen, J. Bayer, S. Urban, and P. van der Smagt, "Efficient movement representation by embedding dynamic movement primitives in deep autoencoders," in *15th IEEE-RAS International Conference on Humanoid Robots*, 2015, pp. 434–440.

[6] A. Colome, G. Neumann, J. Peters, and C. Torras, "Dimensionality reduction for probabilistic movement primitives," in *Proceedings of the International Conference on Humanoid Robots (HUMANOIDS)*, 2014.

[7] H. van Hoof, N. Chen, M. Karl, P. van der Smagt, and J. Peters, "Stable reinforcement learning with autoencoders for tactile and visual data," *IROS*, 2016.

[8] M. Karl, M. Soelch, J. Bayer, and P. van der Smagt, "Deep variational Bayes filters: Unsupervised learning of state space models from raw data," *arXiv:1605.06432*, 2016.

[9] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[10] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," *CoRR*, vol. abs/1312.6114, 2013.

[11] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th International Conference on Machine Learning*, 2008, pp. 1096–1103.

[12] J. Bayer and C. Osendorfer, "Learning stochastic recurrent networks," in *NIPS Workshop on Advances in Variational Inference*, 2014.

[13] J. Chung, K. Kastner, L. Dinh, K. Goel, A. C. Courville, and Y. Bengio, "A recurrent latent variable model for sequential data," in *Advances in neural information processing systems*, 2015, pp. 2980–2988.

[14] M. Sölch, J. Bayer, M. Ludersdorfer, and P. van der Smagt, "Variational inference for on-line anomaly detection in high-dimensional time series," *ICML Workshop*, 2016.

[15] L. Theis, A. v. d. Oord, and M. Bethge, "A note on the evaluation of generative models," *International Conference on Learning Representations*, 2016.

[16] S. Mandt, J. McInerney, F. Abrol, R. Ranganath, and D. Blei, "Variational tempering," in *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, 2016, pp. 704–712.

[17] A. Wilson, A. Fern, S. Ray, and P. Tadepalli, "Multi-task reinforcement learning: a hierarchical bayesian approach," in *Proceedings of the 24th international conference on Machine learning*. ACM, 2007, pp. 1015–1022.

[18] H. Hoffmann, P. Pastor, D.-H. Park, and S. Schaal, "Biologically-inspired dynamical systems for movement generation: automatic real-time goal adaptation and obstacle avoidance," in *IEEE International Conference on Robotics and Automation*, 2009, pp. 2587–2592.

[19] P. I. Corke, *Robotics, Vision & Control: Fundamental Algorithms in MATLAB*. Springer, 2011, iSBN 978-3-642-20143-1.