Proc. 2015 IEEE-RAS International Conference on Humanoid Robots

Efficient movement representation by embedding Dynamic Movement Primitives in Deep Autoencoders

Nutan Chen, Justin Bayer, Sebastian Urban, Patrick van der Smagt

***Abstract*— Predictive modeling of human or humanoid movement becomes increasingly complex as the dimensionality of those movements grows. Dynamic Movement Primitives (DMP) have been shown to be a powerful method of representing such movements, but do not generalize well when used in configuration or task space. To solve this problem we propose a model called *autoencoded dynamic movement primitive (AE-DMP)* which uses deep autoencoders to find a representation of movement in a latent feature space, in which DMP can optimally generalize. The architecture embeds DMP into such an autoencoder and allows the whole to be trained as a unit. To further improve the model for multiple movements, sparsity is added for the feature layer neurons; therefore, various movements can be observed clearly in the feature space.**

After training, the model finds a single hidden neuron from the sparsity that can efficiently generate new movements. Our experiments clearly demonstrate the efficiency of missing data imputation using 50-dimensional human movement data.

I. Introduction

The compact representation of movement sequences is a key element in predicting or analyzing such movement. For voluntary limb movement, simple polynomial approximations may suffice as these can represent minimal acceleration change over time [1]. However, for more complex movements, involving interaction or whole-body motion, such simple models do not suffice; indeed, a closed-form representation can usually not be found.

Dynamic movement primitives (DMP) were developed to represent movement for programming by demonstration [2]. They represent nonlinear dynamic systems by second-order differential equations, if the parameters are correspondingly tuned. Additional parameters allow their variation with respect to their state space, thus varying, for instance, speed of motion. In their original interpretation, DMPs are learned in joint space of the robot or human. This has the disadvantage that adaptation of the DMP parameters has no clear relationship to the task space, and the DMPs do not generalize well. Also, setting the DMPs up in task space [3] is problematic for high-dimensional systems, e.g. for humanoid or human movement. Instead we were inspired by the approach in [4] by representing the movement in a physically not defined, latent, space. In that paper, the authors propose to embed DMPs in Gaussian Process Latent Variable Models (GPLVM) by (a) having the GPLVMs learn a latent space of the movement, and (b) applying DMPs in that latent space.

The authors are with the Faculty for Informatics, Technische Universität München, 80333 Germany nutan(at)in.tum.de, bayer.justin(at)gmail.com, surban(at)tum.de. PvdS is also with fortiss, TUM Associate Institute.

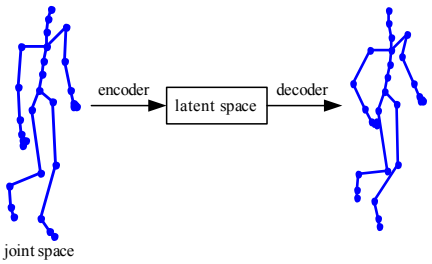


Fig. 1: AE-DMP: Describing movement modified in latent space.

Alternative approaches, building on the representational power of deep neural networks, have obtained competitive results. In [5]–[7] connectionist models were trained to represent human movement; these methods rely on binary latent variables of which it is questionable whether they pose a reasonable prior for human or humanoid kinematics. [8] propose a recurrent architecture with continuous latent variables, but their approach does not exploit prior knowledge on human motion.

In the approach we take in this paper we extend deep neural networks in the form of autoencoders [9] with DMPs. The autoencoder is used to learn a nonlinear dimension reduction of its data, leading to latent representations of these data in a layer of hidden units (as exemplarily illustrated in Fig. 1). However, rather than having these hidden units represent general data, we restrict them to be DMPs which can optimally represent dynamic movement, and generalize to similar movements of the system that is represented. In effect, we constrain the latent representation to represent the dynamics of the investigated system.

In the sequel we will formally define our approach and demonstrate its use on human kinematic data from the CMU Motion Capture Database.

II. Autoencoded Dynamic Movement Primitives

In this section, our new model of deep autoencoders, and specifically, denoising autoencoders, fitted with dynamic movement primitives will be presented. In order to extract the features, a deep autoencoder [9] is used to reduce the dimensionality from instantaneous high-dimensional joint information, while DMPs are able to make use of the feature data for a movement coded in time.

A. Autoencoder

The effect of autoencoder is coarsely comparable to principal component analysis (PCA). Yet, the representational capabilities are much greater since autoencoders allow for non-linear feature extraction. Also, autoencoders are not limited to normally distributed data and do not assume perpendicularity of the "principal components". To fit the human movement data, the activation function and cost function are different from [9].

1) Basic Autoencoder: An autoencoder, which consists of encoder networks and decoder networks, is a neural network method to learn features from unlabeleld data. In our setup, the encoder network takes d joint angles $\mathbf{x} \in [-1,1]^d$ of a moving body as input vector, and maps the input to a latent representation with multiple hidden layers. Every hidden layer computes a mapping $\mathbf{y} = \mathbf{h}_\theta(\mathbf{x}) = \tanh(\mathbf{W}\mathbf{x} + \mathbf{b})$, where $\theta = \{\mathbf{W}, \mathbf{b}\}$ are parameters, $\mathbf{y} \in [-1,1]^{d'}$ is the feature representation, and d' is the number of hidden neurons in that layer.

Subsequently, the feature representation is reconstructed back to a joint vector $\mathbf{z} \in [-1,1]^d$ through the decoder networks with the same structure. The decoder hidden layers are $\mathbf{z} = g_{\theta'}(\mathbf{y}) = \tanh(\mathbf{W}'\mathbf{y}+\mathbf{b}')$, where $\theta' = \{\mathbf{W}', \mathbf{b}'\}$, $\mathbf{b}'$ is a bias vector in the decoder layers, and the weight matrix is restricted to be equal to the transpose of the encoder weights $\mathbf{W}' = \mathbf{W}^T$. To seek the parameters θ and θ', the problem becomes:

$$\begin{aligned}\theta^\star, \theta'^\star &= \arg\min_\theta \frac{1}{n}\sum_{i=1}^{n} L(\mathbf{x}^{(i)}, \mathbf{z}^{(i)}) \\ &= \arg\min \frac{1}{n}\sum_{i=1}^{n} L\Big[\mathbf{x}^{(i)}, g_{\theta'}\big(h_\theta(\mathbf{x}^{(i)})\big)\Big], \end{aligned} \tag{1}$$

where n is the number of a training set, L is a loss function and, assuming Gaussian noise, the squared error $L(\mathbf{x}, \mathbf{z}) = \|\mathbf{x} - \mathbf{z}\|^2$ is used to represent the error in the joint data.

The number of neurons in the hidden layer is less than that of the input layer, which pushes the data through a bottleneck and forces it to extract the most relevant features. It also makes the trivial solution of the identity function at each neuron impossible.

2) Denoising: Denoising autoencoders are based on the basic autoencoder with corruption of the input during the training process. The model is able to robustly reconstruct undestroyed human joints from a partially corrupted one. We partially destroy the initial $\mathbf{x}$ to generate $\tilde{\mathbf{x}}$ as the input instead. For every input frame $\mathbf{x}_i$, each of the inputs joint is picked with a probability of p, and its value is set to 0, while the rest are unchanged. Thus, the chosen joints are dropped, and we have

$$\theta^\star, \theta'^\star = \underset{\theta,\theta'}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n} L\Big[\mathbf{x}^{(i)}, g_{\theta'}\big(h_\theta(\tilde{\mathbf{x}}^{(i)})\big)\Big]. \tag{2}$$

The layers are fully connected except the removed neurons.

During testing process, all joints are present without corruption and the weights of the first input layer to the first encoder layer are scaled by multiplying $1-p$. The weights on other layers and bias are unchanged.

B. Dynamic Movement Primitives

DMPs are generally trained from a demonstration of a trajectory in its feature space, which can then subsequently be reproduced and adjusted [10]. A DMP is a point attractor using a second-order dynamic system:

$$\tau\ddot{\mathbf{y}} = \alpha_z(\beta_z(\mathbf{g} - \mathbf{y}) - \dot{\mathbf{y}}) + \mathbf{f}, \tag{3}$$

where τ is a time constant and α_z and β_z are damping constants. In our setting, instead of representing joint angles, we use the features $\mathbf{y}$ to represent the trajectory in latent space. The difference term $(\mathbf{g} - \mathbf{y})$ attracts the trajectory to the goal position $\mathbf{g}$, and we set the the final frame of the demonstration as the goal.

Following [10], we can choose the the forcing term $\mathbf{f}$ as a linear combination of basis functions Ψ_i:

$$\mathbf{f}(t) = \frac{\sum_{i=1}^{N}\Psi_i(t)\mathbf{w}_i}{\sum_{i=1}^{N}\Psi_i(t)}, \tag{4}$$

to create a linear time-variant dynamical system. To decouple this from the dynamics of the data, the time t is replaced using a first-order linear dynamic system

$$\tau\dot{s} = -\alpha_s s, \tag{5}$$

where α_s is a constant coefficient. Thus the state s converges monotonically to zero.

By differing the basis functions Ψ we can let DMPs be discrete or rhythmic dynamical systems. We focus on the former but an extension to rhythmic dynamical systems is straightforward. Ψ are written as [10]

$$\Psi_i(s) = \exp\Big[-\frac{1}{2\sigma_i^2}(s - c_i)^2\Big], \tag{6}$$

where c_i and σ_i describe the width and centers of the basis functions, respectively.

In the training process of our AE-DMP, with features $\mathbf{y}_{\text{demo}}$ extracted from the autoencoder and its derivatives computed, we obtain the target values of the forcing term,

$$\mathbf{f}_t = \tau^2\ddot{\mathbf{y}}_{\text{demo}} - \alpha_z(\beta_z(\mathbf{g} - \mathbf{y}_{\text{demo}}) - \tau\dot{\mathbf{y}}_{\text{demo}}). \tag{7}$$

With $\{(\mathbf{f}_t^{(1)}, \mathbf{f}(s)^{(1)}), \ldots, (\mathbf{f}_t^{(m)}, \mathbf{f}(s)^{(m)})\}$ of length m, the corresponding solution is derived as

$$\mathbf{w}^\star = \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{i=1}^{m} L\big[\mathbf{f}_t^{(i)}, \mathbf{f}(s)^{(i)}\big]. \tag{8}$$

C. Autoencoded Dynamic Movement Primitive

Our model imbeds DMP into denoising autoencoder seamlessly by taking DMP as one of the hidden layer as shown in Fig. 2. The amount of hidden layers can be increased to form deeper architectures.

The parameters $\mathbf{w}$ can be trained using various machine learning methods such as locally weighted regression (LWR) [11] and Gaussian Mixture Models (GMM) [12]. However,

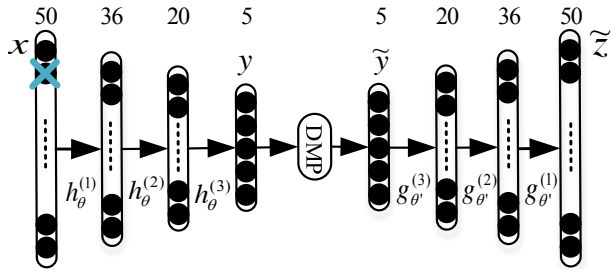


Fig. 2: Architecture of AE-DMP. $\times$ represents the joints removed by denoising. Numbers indicate hidden layer sizes.

in order to fit DMP into autoencoder, the learning of $\mathbf{w}$ is accomplished using neural networks.

Therefore, the minimizer of AE-DMP is rewritten from (2) and (8) as

$$\theta^\star, \theta'^\star, \mathbf{w}^\star = \underset{\mathbf{w},\theta,\theta'}{\operatorname{argmin}} \sum_{i=1}^{m} \Big\{ L(\mathbf{x}^{(i)}, \tilde{\mathbf{z}}^{(i)}) + \lambda L\Big[\mathbf{f}_t^{(i)}, \mathbf{f}(s)^{(i)}\Big] \Big\}, \tag{9}$$

where λ is a penalty parameter, and different from (2) $\tilde{\mathbf{z}}$ is the reconstruction of the human movement in joint space based on the output of the DMP component $\tilde{\mathbf{y}}$. The second term of the the penalty $L(\mathbf{f}_t^{(i)}, \mathbf{f}(s)^{(i)})$ enables the DMP output to follow the demonstration accurately in the feature space. In addition, without $L(\mathbf{f}_t^{(i)}, \mathbf{f}(s)^{(i)})$, the neural networks take Ψ as the input and the encoder layers are missing during the training process. The weights are updated for a whole sequence batch. We have one $\mathbf{w}$ for one demonstration, because DMP is a model trained by one demonstration.

We pre-train the model by denoising autoencoder and DMP separately, then obtain the final weights from (9). Thus, the model autonomously learns movements in the feature space while without hand-engineered representations.

D. Sparse AE-DMP

When the data contains multiple motions, AE-DMP is able to switch the motions or interpolate novel motions between existing ones. Sparse AE-DMP is an effective approach to achieve this with reasonable explanation in the hidden units. Sparsity encourages hidden units to be active only rarely [13] by adding an extra term of l_1 norm penalty to the cost

$$L(\mathbf{y}) = \eta \sum_{i=1}^{m} \|\mathbf{y}^{(i)}\|_1, \tag{10}$$

where η is a penalty parameter, and the l_k norm with $k \geq 1$ is defined as $\|\mathbf{y}\|_k \triangleq (\sum_{i=1}^{d'} |y_i|^k)^{1/k}$. It results in a sparse solution for $\mathbf{y}$. For various movements, sparsity is designed to deactivate a part of hidden neurons for distinguishing the movements, so l_1 norm which can yield extreme small values of $\mathbf{y}$ is chosen. On the other hand, l_2 cannot deactivate the neurons and the derivation of pseudo-norm l_0 is not smooth.

Thus, for multiple movements, sparsity encodes one movement into only a few hidden units while other movements into other units. To switch or interpolate existing movements, we can simply deactivate/activate or interpolate these hidden units. This approach can be used as a motion primitive library for further development such as movement segmentation and recognition.

III. Experiments

A. Human Motion Data

We use the CMU Graphics Lab Motion Capture Database to evaluate AE-DMP. That database is created by tracking 41 markers on human subjects during walking, jogging, etc. using a Vicon optical tracking system. The data are preprocessed using Vicon Bodybuilder to render limb joint angles, leading to a 62-dimensional feature vector. Of these, 6 values are almost constant (shoulder and finger movement, having an insignificant variance $< 10^{-26}$) and are ignored. Another set of 6 values represent the origin frame, and can be removed after all data are represented with respect to this frame. The resulting feature vector used in this paper is 50-dimensional. For the purpose of this paper we evaluated results on 5 subjects (viz. Subject 35, 49, 74, 120 and 143) with various movements.

Before training to the AE-DMP, the data is normalized in every dimension to zero mean and range in $[-1, 1]$. Visualization is done by means of the software described in [14].

In the following experiments, we use the mean square error (MSE $\pm$ standard deviation) to numerically evaluate the model. All results are reported in radians. A video of the results is available at http://brml.org/fileadmin/videos/AEDMP.mp4.

B. Features in the Hidden Neurons

To illustrate what movement the hidden neurons in AE-DMP encode, we train the model using a walking motion in this experiment. 21 sequences are trained and every sequence is segmented or scaled to 70 frames. After that, additional results of 5 subjects with 1 sequence for every movement are evaluated. We use the autoencoder to reduce the input dimensionality from 50 to 5. The used autoencoder consists of one input layer, 3 encoder layers with 36, 20 and 5 neurons respectively, one dynamic movement primitive layer, and 3 decoder layers plus output layer (see Fig. 2). The number of neurons in every layer is chosen for minimization of the reconstruction error for autoencoder. In the DMPs we use 20 weighting kernels Ψ of the primitive. Following [10] we set $\beta_z = 0.25\alpha_z$ to obtain critical damping. β_z can be tuned to vary the movement between smoothness and accuracy of following the given trajectory.

Fig. 3 shows the data distribution in two of the five hidden neurons after the DMP component. The distribution is plotted in those two which have the two largest variance.

A movement of walking in feature space is illustrated in Fig. 4 for two hidden neurons; other hiddens render similar images. Hidden neuron 2 has the largest variance among all of the hidden neurons while hidden neuron 3 has the smallest. The DMP component in the AE-DMP is able to model the motions in the hidden space. The mean square error of the experiment in the joint space is $3.8 \cdot 10^{-4} \pm 1.2 \cdot 10^{-3}$ rad.

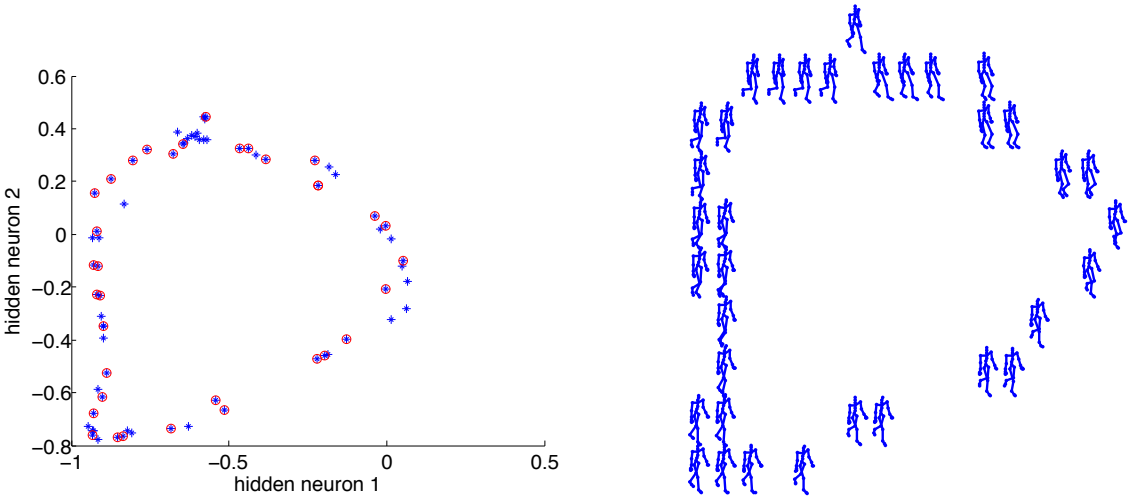
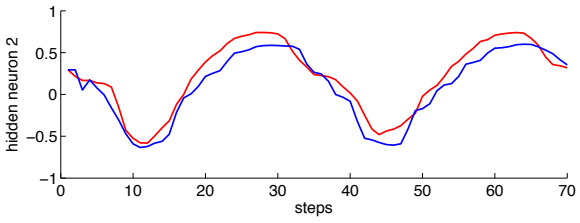
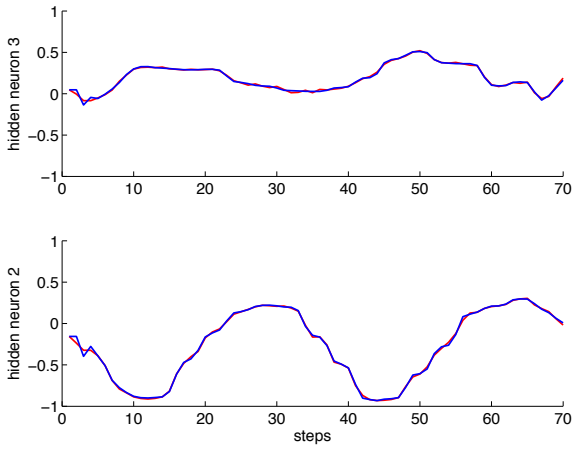


Fig. 3: Data in hidden neurons of integrated AE-DMP. The blue stars in the left figure are a sequence data in the two hidden neurons, and the red circles in the left figure are selected to be plotted in the right figure in the joint space. For visualization, the selected points probably are shifted to a grid in the right figure. The front leg of the human model smoothly changes from the left leg to be the right leg in the right figure from the top to down. Additionally, a walking movement can be seen in a counterclockwise direction in the right figure.



(a): no regularization



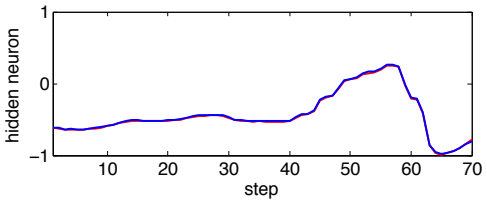
(b): with regularization

Fig. 4: AE-DMP trajectory in the feature space. The red line, extracted from the autoencoder, is the ground truth of the DMP, and the blue line is the DMP prediction of the trajectory. In the top figure, the DMP prediction without regularization is given. For the bottom two figures, the regularization term $\mathbf{f}$ is added.

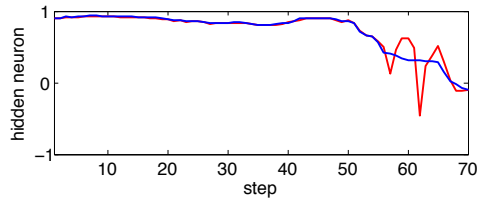
Fig. 4 demonstrates the effect of the regularization term $L(\mathbf{f}_t^{(i)}, \mathbf{f}(s)^{(i)})$. In Fig. 4(a), the autoencoded input to the DMP is given (red line) as well as the output of the DMP, without using this regularization term. In addition, Fig. 4 shows the improved prediction when adding $L(\mathbf{f}_t^{(i)}, \mathbf{f}(s)^{(i)})$.

A major strength of our method is caused by integrating the dimensionality reduction with the DMP-based movement representation. Other approaches, such as [4], separate these steps by first doing a dimensionality reduction and then representing the resulting latent space in DMPs. To demonstrate the advantage of our integrated approach we represented the 5 subjects with different movements (a) in our AE-DMP model, and (b) by first running an Autoencoder-based dimensionality reduction, and then using DMPs in the resulting latent space. The results are numerically reported in Table I. The same movement results are shown in Fig. 6 with more details in every joint. For simple, static movement movements (in particular, balancing), the non-integrated model compares to the integrated AE-DMP. However, for movements, and esp. jerk movements such as kicking and complicated movement such as tai chi which consists of several different movements, show a considerable improvement of AE-DMP over the non-integrated model. In AE-DMP, the constraints that the DMP impose on the latent representation force the autoencoder to smoothen the hidden values, leading to much improved results. For instances, Fig. 5 illustrates the jerk trajectory of kicking in the hidden space using non-integrated model, but the trajectory is smooth by the integrated model.

Compared with DMP and state of the art PCA dimensionality reduction for DMP, AE-DMP is more accurate as shown in Table I. Similar as non-integrated method, both DMP and PCA-DMP have difficulty of jerk movement,



(a): Integrated AE-DMP. Other 4 trajectories of hidden neurons are as smooth as the illustrated trajectory.



(b): Non-integrated AE-DMP. 2 out of other 4 hidden neurons have jerk trajectories as the illustrated trajectory.

Fig. 5: AE-DMP trajectory of kicking movement in the feature space. The red line, extracted from the autoencoder, is the ground truth of the DMP, and the blue line is the DMP prediction of the trajectory.

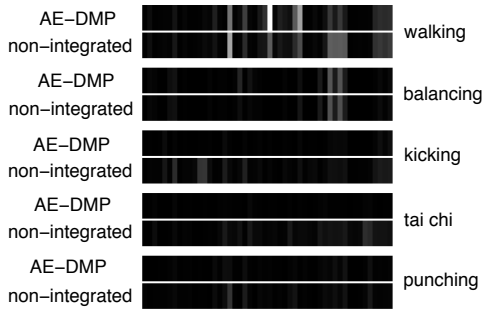


Fig. 6: R-squared accuracy ($\sqrt{R^2}$). The R-squared value is indicated by a gray value, with black for 1 and white for 0. The horizontal axis represents 50 joints.

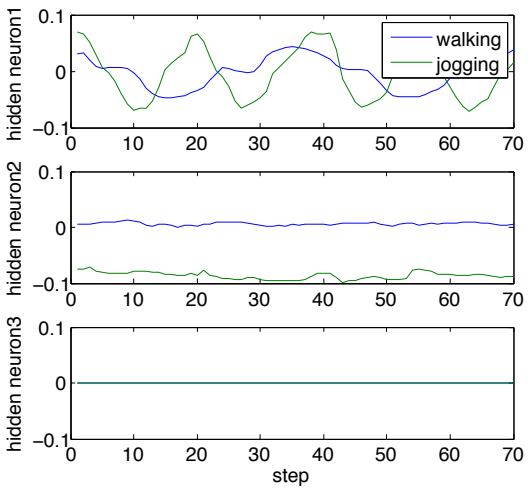


Fig. 7: Feature representation of sparse AE-DMP. There are 3 kinds of hidden neurons for multiple movements. Both of walking and jogging are active as shown in the top figure, which encodes the information such as frequency of the movement; jogging is active while walking is not active as shown in the middle figure, which encodes the gesture of the movement; both of walking and jogging are not active as shown in the bottom figure.

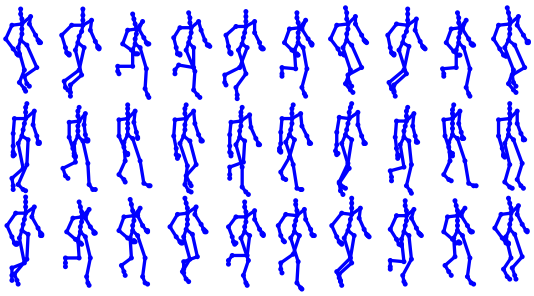


Fig. 8: Novel motion generation. The top two rows are the training data of jogging and walking. The bottom row shows jogging generation from walking. It is a time sequence from left to right and plotted every 6 steps.

and PCA-DMP lose some information of the movements during dimensionality reduction. For comparison, all the three model use the same parameters for DMP and PCA reduces dimensions to 5.

C. New Motion Generation

In this experiment, the walking movement and jogging movement are trained in one model. The training data includes 21 sequences of walking and 9 sequences of jogging.

Using sparse AE-DMP, there are 3 types of hidden values for multiple movements as shown in Fig. 7. By activating, deactivating or interpolating though hidden neuron 2, new movements are generated. As an example, Fig. 8 illustrates new movement generation from walking by activating the hidden neuron 2. The generated movement follows the walking demo in the terms of rhythm, while only the gesture becomes to jogging; therefore, the generated movement is slow jogging which has never shown in the training data. The interpolation of neuron 2 based on the walking movement has new motions between the middle and bottom of Fig. 8. Because we do not have ground truth of the new motion, it is not numerically evaluated.

The basic AE-DMP without sparsity can also generate or interpolate new motions from existing movements, but it is relative complicated because the hidden neurons which distinguish the motions may have intersection. Compared with the basic AE-DMP, it simplifies the problem to single hidden neuron modification using sparsity.

D. Reconstruction for Missing Joints

To illustrate the validity of our method, we demonstrate missing value imputation of missing joints and missing sub sequences. The DMP component is not used in for joint reconstruction—this will be part of the next section.

TABLE I: Error of the movement modeling, averaged over all joints over a whole movement. The error is MSE $\pm$ SD of the joint angle in radians. The AE-DMP is superior in all cases, but in particular for highly dynamic movements.

movement	walking	balance	kicking	tai chi	punching
subject #	35	49	74	120	143
AE-DMP	$3.3{\cdot}10^{-4} \pm 1.2{\cdot}10^{-3}$	$2.8{\cdot}10^{-4} \pm 1.7{\cdot}10^{-3}$	$2.3{\cdot}10^{-4} \pm 1.7{\cdot}10^{-3}$	$3.4{\cdot}10^{-4} \pm 1.7{\cdot}10^{-3}$	$1.5{\cdot}10^{-4} \pm 0.93{\cdot}10^{-3}$
non-integrated	$5.1{\cdot}10^{-4} \pm 2.4{\cdot}10^{-3}$	$2.9{\cdot}10^{-4} \pm 1.8{\cdot}10^{-3}$	$21 \cdot 10^{-4} \pm 30 \cdot 10^{-3}$	$25 \cdot 10^{-4} \pm 12 \cdot 10^{-3}$	$2.4{\cdot}10^{-4} \pm 1.7{\cdot}10^{-3}$
DMP	$5.6{\cdot}10^{-4} \pm 2.5{\cdot}10^{-3}$	$2.7{\cdot}10^{-4} \pm 1.8{\cdot}10^{-3}$	$20 \cdot 10^{-4} \pm 27 \cdot 10^{-3}$	$26 \cdot 10^{-4} \pm 13 \cdot 10^{-3}$	$8.4{\cdot}10^{-4} \pm 4.4{\cdot}10^{-3}$
PCA-DMP	$11 \cdot 10^{-4} \pm 4.0{\cdot}10^{-3}$	$6.0{\cdot}10^{-4} \pm 2.4{\cdot}10^{-3}$	$45 \cdot 10^{-4} \pm 28 \cdot 10^{-3}$	$38 \cdot 10^{-4} \pm 14 \cdot 10^{-3}$	$17 \cdot 10^{-4} \pm 57 \cdot 10^{-3}$

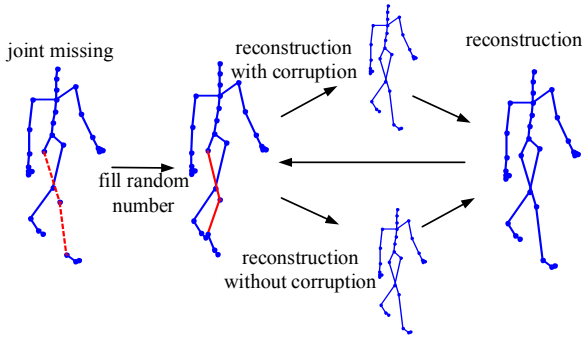


Fig. 9: Missing joint reconstruction process. The red dashed lines are the ground truth of the missing joints with the corresponding limbs and the red solid lines are missing joints with corresponding limbs during reconstruction process.

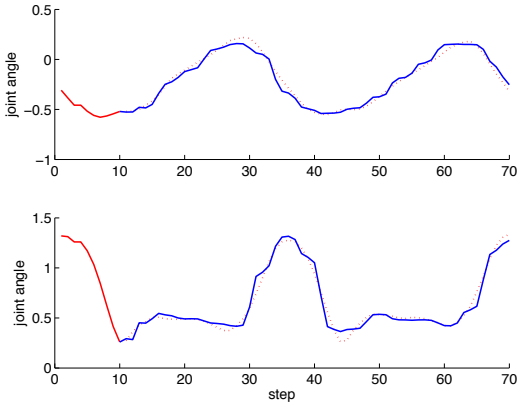


Fig. 10: Filling data for missing section. The top and bottom figures are the hip pitch and knee pitch of the right leg respectively. The red solid line is the demo motion of testing data, the red dotted line is the truth of the missing section, and the blue solid line is the data filled using AE-DMP.

Denoising autoencoders have been shown to be generative models [15]: it turns out that corrupting an input and feeding it through the denoising autoencoder performs a step in a Markov chain, of which the equilibrium distribution is equal to the data generating distribution. This enables a principled missing value imputation method.

Let $\mathbf{x}_J$ denote some the observed parts of the input $\mathbf{x}$; conversely, let $\mathbf{x}_{\bar{J}}$ denote the non-observed parts of the input. To find a reasonable imputation $\hat{\mathbf{x}}$, we can thus pass $\mathbf{x}_J \cup \mathbf{x}_{\bar{J}}$ through the autoencoder to obtain $\mathbf{x}_J^{(1)}$ and $\mathbf{x}_{\bar{J}}^{(1)}$. Here, the former is not corrupted, necessitating an adaption of the weights by a factor of $1-p$. $\mathbf{x}_J^{(1)}$ is henceforth discarded, and $\mathbf{x}_J \cup \mathbf{x}_{\bar{J}}^{(1)}$ passed through the network. This process is repeated to give $\mathbf{x}_{\bar{J}}^{(k)}$, which will be an unbiased sample of $p(\mathbf{x}_{\bar{J}}|\mathbf{x}_J)$. We then use $\hat{\mathbf{x}} = \mathbf{x}_J \cup \mathbf{x}_{\bar{J}}^{(k)}$ as the datum with imputed values. For our experiments, the chain converged after $k = 4$ steps. See Fig. 9 for an illustration.

In the experiments, we removed four right leg joints or four left arm joints out of fifty joints to generate two time sequences of testing data. Notably, a denoising autoencoder can reconstruct the missing joints in one frame without the time sequence information. The reconstruction error in joint space is shown in Table II.

E. Reconstruction for Missing Section

We train the model as III-B, and the testing data has a demo of the first 10 frames while the last 60 frames are missing. Fig. 10 shows the reconstruction for the hip pitch and knee pitch of the right leg while other joints have similar plot. The error is $8.1{\cdot}10^{-4} \pm 3.4{\cdot}10^{-3}$ rad.

F. Changing Goal Attractor

AE-DMP maintains the generalization properties of DMP. Taking punching movement as an instance, Fig. 11 shows generalization with changing the attractor landscape. All of the start points are the same, while the testing movements are generalized with the shifted goals. The result is shown in two largest variance neurons. The generalized movement in joint space is shown in the video.

In this punching movement, it does not require scaling properties or invariance properties. For other movements such as writing which need invariance properties, the right side of (4) can be easily multiply by $(\mathbf{g} - \mathbf{y}_0)$ [10].

IV. Conclusions

In their standard formulation, DMPs suffer from suboptimal generalization—when used in configuration space—or are a victim of the curse of dimensionality—when used in task space. This is especially true for high-dimensional data sets, e.g. while representing the movement of human(oid) limbs. We present AE-DMP, a model that integrates DMPs with autoencoders to perform dimensionality reduction while optimally representing movement in latent feature space. The

TABLE II: Result of reconstruction for missing joints in joint space. The error is MSE $\pm$ SD of the joint angle in radians.

joint name	hip pitch	hip yaw	hip roll	knee pitch
error	$1.5{\cdot}10^{-3} \pm 2.0{\cdot}10^{-3}$	$1.1{\cdot}10^{-3} \pm 1.3{\cdot}10^{-3}$	$0.33{\cdot}10^{-3} \pm 0.50{\cdot}10^{-3}$	$5.0{\cdot}10^{-3} \pm 8.5{\cdot}10^{-3}$

joint name	shoulder pitch	shoulder yaw	shoulder roll	elbow pitch
error	$0.74{\cdot}10^{-3} \pm 0.91{\cdot}10^{-3}$	$0.73{\cdot}10^{-3} \pm 0.97{\cdot}10^{-3}$	$0.38{\cdot}10^{-3} \pm 0.048{\cdot}10^{-3}$	$2.7{\cdot}10^{-3} \pm 3.6{\cdot}10^{-3}$

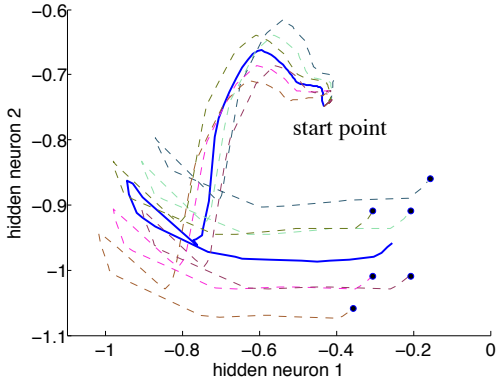


Fig. 11: Changing goal attractor of punching movement in latent space. The solid line is the demonstration, the dashed line is the generalization movements with various goal attractors, and the black circles are the shifted goals.

integration is shown to create a movement feature space in which DMPs can efficiently code and generalize movement.

To elaborate the connections and difference of various types of movements in feature space, we develop AE-DMP using sparsity. Sparse AE-DMP (SAE-DMP) can generate new movements which are not in the training data set by simply switching on/off or interpolating one hidden neuron. The new movement combines features of the existing two. For instance, slow jogging is generated based on walking and switching the movement type to jogging.

Experiments on a public motion capture database demonstrate that the model can accurately fill in corrupted data. The autoencoder facilitates the reconstruction of missing joints, while the DMP represents the dynamics to reconstruct missing movement sections.

In subsequent work we aim to integrate DMPs with variational autoencoders [16] (VAE) to obtain a probabilistic version of VA-DMP. It allows us to use the DMP dynamics for sampling the latent variables in the VAE, and thus steer the movement. Since VAE maximize the probabilistic independence in latent space, it will lead to a representation where the DMPs represent statistically independent parts of the movement, thus maximizing the generalizability.

V. Acknowledgements

This work has been supported in part by the TACMAN project, EC Grant agreement no. 610967, within the FP7 framework programme. The data used in this project was obtained from mocap.cs.cmu.edu. The database was created with funding from NSF EIA-0196217.

References

[1] T. Flash and N. Hogan, "The coordination of arm movements: an experimentally confirmed mathematical model," *The journal of Neuroscience*, vol. 5, no. 7, pp. 1688–1703, 1985.

[2] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," in *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, vol. 2. IEEE, 2002, pp. 1398–1403.

[3] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, "Learning and generalization of motor skills by learning from demonstration," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, 2009, pp. 763–768.

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

[5] G. W. Taylor, G. E. Hinton, and S. T. Roweis, "Modeling human motion using binary latent variables," in *Advances in Neural Information Processing Systems 19*, B. Schölkopf, J. Platt, and T. Hoffman, Eds. MIT Press, 2007, pp. 1345–1352.

[6] I. Sutskever, G. E. Hinton, and G. W. Taylor, "The recurrent temporal restricted boltzmann machine," in *Advances in Neural Information Processing Systems*, 2009, pp. 1601–1608.

[7] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent, "Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription," *arXiv preprint arXiv:1206.6392*, 2012.

[8] J. Bayer and C. Osendorfer, "Learning stochastic recurrent networks," *arXiv preprint arXiv:1411.7610*, 2014.

[9] 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*, ser. ICML '08. New York, NY, USA: ACM, 2008, pp. 1096–1103.

[10] 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, February 2013.

[11] S. Schaal and C. G. Atkeson, "Constructive incremental learning from only local information," *Neural Comput.*, vol. 10, no. 8, pp. 2047–2084, Nov. 1998.

[12] S. Calinon, Z. Li, T. Alizadeh, N. G. Tsagarakis, and D. G. Caldwell, "Statistical dynamical systems for skills acquisition in humanoids," in *Proc. IEEE Intl Conf. on Humanoid Robots (Humanoids)*, Osaka, Japan, 2012.

[13] V. Nair and G. E. Hinton, "3d object recognition with deep belief nets," in *Advances in Neural Information Processing Systems 22*. Curran Associates, Inc., 2009, pp. 1339–1347.

[14] R. Urtasun, D. J. Fleet, A. Geiger, J. Popovic, T. Darrell, and N. D. Lawrence, "Topologically-constrained latent variable models." in *ICML*, ser. ACM International Conference Proceeding Series, vol. 307, 2008, pp. 1080–1087.

[15] Y. Bengio, L. Yao, G. Alain, and P. Vincent, "Generalized denoising auto-encoders as generative models," in *Advances in Neural Information Processing Systems*, 2013, pp. 899–907.

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