Efficient Nonlinear Markov Models for Human Motion

Andreas M. Lehrmann
MPI for Intelligent Systems
Tuebingen, Germany
alehrmann@tue.mpg.de

Peter V. Gehler
MPI for Intelligent Systems
Tuebingen, Germany
pgehler@tue.mpg.de

Sebastian Nowozin
Microsoft Research
Cambridge, UK
sebastian.nowozin@microsoft.com

Abstract

Dynamic Bayesian networks such as Hidden Markov Models (HMMs) are successfully used as probabilistic models for human motion. The use of hidden variables makes them expressive models, but inference is only approximate and requires procedures such as particle filters or Markov chain Monte Carlo methods. In this work we propose to instead use simple Markov models that only model observed quantities. We retain a highly expressive dynamic model by using interactions that are nonlinear and non-parametric. A presentation of our approach in terms of latent variables shows logarithmic growth for the computation of exact log-likelihoods in the number of latent states. We validate our model on human motion capture data and demonstrate state-of-the-art performance on action recognition and motion completion tasks.

1. Introduction

Statistical models for human motion are important in many areas of computer vision and graphics. In addition to being interesting in their own right [29, 26, 22], their applications include areas as diverse as animation, robotics, tracking [25, 27], and activity recognition [4]. While great progress has been made in the past decade, the problem remains challenging because of the high dimensionality, nonlinearity and multimodality of natural human motion. Ideally, a good probabilistic model should account for all of those challenges, but unfortunately expressive models often result in intractable estimation and inference problems. We now review the most popular approaches.

1.1. Dynamic Bayesian Networks

In recent years, modelling of human motion has been tackled from several different perspectives. Among the most popular methods are latent variable models following the state-space equations (see also Figure 2a),

$$z_t = f(z_{t-1}, \epsilon_z) \quad \leftrightarrow \quad p_z(z_t \mid z_{t-1}), \tag{1}$$
$$x_t = g(z_t, \epsilon_x) \quad \leftrightarrow \quad p_x(x_t \mid z_t), \tag{2}$$

where x_t are observable variables such as joint positions or joint angles, z_t are hidden variables, and ϵ_x, ϵ_z are random perturbations. Although filtering and smoothing distributions for z_t are within this framework, they are only tractable for either a discrete state space (forward(-backward) algorithm) or for linear functions f, g and additive Gaussian noise (Kalman filter/smoother) [14]. Efficient and exact solutions for the general nonlinear and/or non-Gaussian cases do not exist [8]. To perform inference we therefore need to resort to approximate methods, such as the extended Kalman filter and its derivates [19, 18], or to Sequential Monte Carlo methods like particle filters [15, 8, 7]. Augmenting the state-space model with discrete switching variables leads to a Switching Linear Dynamical System (SLDS). Exact inference in this model is intractable even for the Gaussian-linear case [2] and learning has turned out to be a challenge in itself [13].

Equations (1–2) and their extensions form the basis for a number of statistical models for human motion: Wang *et al*. [29] assume a linear combination of nonlinear basis functions for f, g and additive Gaussian noise for ϵ_x, ϵ_z. Marginalizing over f, g leads to a Gaussian Process Dynamical Model (GPDM), whose latent trajectories have to be learned by a combination of Scaled Conjugate Gradient and a version of EM using Hybrid Monte Carlo techniques. Urtasun *et al*. [28] extend the GPDM framework by introducing a prior for latent positions that preserves local topological structure. Pavlović *et al*. [22] use approximate variational inference to learn an SLDS and perform inference in it. Taylor *et al*. [26] consider a latent space consisting of binary variables and use a conditional RBM to model human motion. While inference tasks are easy in this model, learning relies on approximations like contrastive divergence [16]. Models inspired by physics and biology include [30] and [31].

1.2. Contributions

In this work we propose to leave out the latent space altogether. Instead we model human motion by means of an expressive Markov model that is both simple enough to al-

low for efficient and exact inference yet flexible enough to accurately model real human motion data. Due to our non-parametric representation, we achieve a high level of realism. Specifically, our work makes the following four contributions: 1. We introduce the Dynamic Forest Model (DFM) and describe its training and regularization, building upon work on autoregressive trees [20]; 2. We present a formulation of our approach in terms of latent variables, thereby allowing a direct comparison to Hidden Markov Models; 3. We show how DFMs can be used as accurate and efficient models of human motion data; 4. We empirically validate DFMs on challenging action recognition and motion completion tasks, outperforming both HMMs and GPDMs.

2. Nonlinear Markov Models

In this section we present our model and its training procedure. A *Markov model* describes a conditional distribution of the present state x_t given a limited number of past observations $\mathrm{pa}(x_t) := x_{(t-K):(t-1)}$. That is, at each time step t a fixed number of K previous observations are combined to form a prediction x_t. The prediction is then an order-K Markov process,

$$p\left(x_t \mid \mathrm{pa}\left(x_t\right)\right) = p\left(x_t \mid x_{(t-K):(t-1)}\right). \tag{3}$$

If the mean of this distribution can be written as a fixed linear combination of the previous observations, the Markov model is said to be linear [11]. When this is not the case, the model is a *nonlinear* Markov model.

2.1. Autoregressive Trees

Autoregressive (AR) trees [20] are a type of probabilistic AR model for time-series data [11] in which the regression function is given by a decision tree.

To represent the distribution in equation (3) the tree evaluates features $\phi\left(\mathrm{pa}\left(x_t\right)\right) \in \mathbb{R}^F$ extracted from the previous K frames and, using this information, decides among a set of simpler distributions stored at its leaf nodes. This is illustrated in Figure 1. We store in each leaf ℓ one multivariate normal distribution with linearly regressed mean but fixed covariance matrix. In this case, the predictive expectation can be obtained by the linear dynamics

$$\mathbb{E}\left[x_t \mid \mathrm{pa}\left(x_t\right)\right] = A_\ell \phi\left(\mathrm{pa}\left(x_t\right)\right).$$

Although this is a simple linear prediction, the choice of which matrix A_ℓ is used is governed by the decision tree and thus a function of $\phi\left(\mathrm{pa}\left(x_t\right)\right)$.[1] For example, by testing for statistics such as average joint velocities in the past K frames, the tree may easily distinguish a running from a walking motion, and hence is able to select the appropriate linear dynamics.

[1] In general one can use different features for linear prediction and leaf node selection.

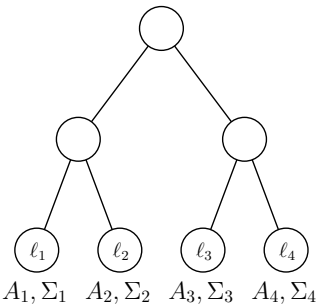


Figure 1: Autoregressive tree. A decision tree is evaluated on a set of features extracted from K previously observed frames, $\phi\left(\mathrm{pa}\left(x_t\right)\right)$. At each leaf ℓ_i of the tree a linear autoregressive model is stored. If leaf ℓ_i is reached, the predictive filtering distribution is defined as $p\left(x_t \mid \mathrm{pa}\left(x_t\right)\right) = \mathcal{N}(A_{\ell_i}\phi\left(\mathrm{pa}\left(x_t\right)\right), \Sigma_{\ell_i})$.

In the original work on autoregressive trees a single tree is learned by greedily optimizing a penalized likelihood objective [20]. The authors show applications to short-term forecasting of univariate economic data but note that 95% of their trees do not contain any splits, i.e., they are common AR models. Here, we propose extensions to AR trees that enable us to take advantage of deeper trees and to cope with high-dimensional inputs, eventually allowing their use for classification, synthesis and upscaling of complex human motion data.

2.2. Dynamic Forests

Because a single tree is prone to overfitting and limited in its expressiveness due to its unimodal predictive distribution, we will instead learn an ensemble of $C > 1$ trees, $\{\mathcal{T}_c\}_{c=1,\ldots,C}$. Each tree is trained separately using bagging [5], that is, resampling the training set framewise with replacement. The predictions of the individual trees are then *averaged* to produce the prediction of the forest. Since each tree has a Gaussian posterior, the forest posterior is given by a multimodal mixture of Gaussians,

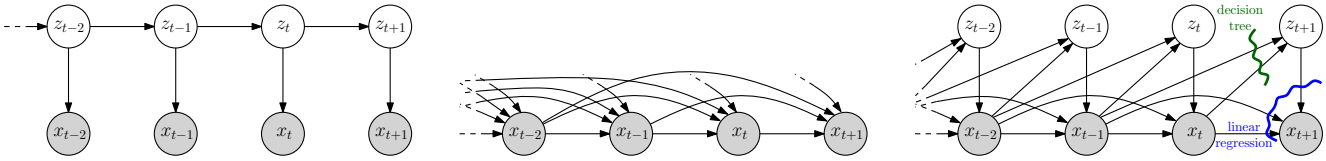
$$p\left(x_t \mid \mathrm{pa}\left(x_t\right)\right) = \frac{1}{C}\sum_{c=1}^{C} \mathcal{N}\left(x_t \middle| \mu_{\ell(c,t)}, \Sigma_{\ell(c,t)}\right).$$

Here, $\ell(c,t) := \ell(c, \phi(\mathrm{pa}(x_t)))$ denotes the leaf node that is selected by the c'th tree at the t'th time step and each mixture component has the mean vector $\mu_{\ell(c,t)} := A_{\ell(c,t)}\,\phi(\mathrm{pa}(x_t))$.

We call this new approach a *Dynamic Forest Model (DFM)* and continue with a description of its training.

2.3. Training

We provide a bottom-up description of the training procedure, i.e., we start with the estimation of a leaf model and then consider the task of learning the tree structure.



(a) Hidden Markov Model. (b) Marginalized Hidden Markov Model. (c) Latent state formulation of DFMs.

Figure 2: **(a)** Discrete time Hidden Markov Models represent a probability distribution of a sequence of observations $(x_t)_t$ by a Markov model on a sequence of hidden variables z_t and a conditional observation distribution $p(x_t|z_t)$. **(b)** Marginalization over the hidden variables yields a joint distribution $p(x_{1:T})$ over the observed variables, effectively coupling all variables. **(c)** Latent space formulation of our proposed nonlinear Markov model for order $K = 2$. A decision tree implicitly selects a latent state and we can view the nonlinear Markov model as an order-K approximation to **(b)**, in which filtering inference and computation of log-likelihoods of observed sequences is very efficient.

The general setup is as follows: At training time we are given a set of N sequences, $\mathbf{D} = (X_i)_{i=1,\dots,N}$. Each sequence X_i is a concatenation of T_i frames, hence $X_i = (x_t^{(i)})_{t=1,\dots,T_i}$. The t'th frame of the i'th sequence is represented by a fixed-length vector $x_t^{(i)} \in \mathbb{R}^d$. We will often drop the sequence index i to keep the notation uncluttered.

2.3.1 Training the Leaf Model

Learning a model for a leaf node ℓ amounts to estimating a regression matrix A_ℓ and a covariance matrix Σ_ℓ. Each leaf accommodates a subset $\{(\phi(\mathrm{pa}(x_t)), x_t)\}_{t\in\mathbf{D}_\ell}$ of the training data, namely those feature vectors and regression targets that reach it. We use the available data points to estimate A_ℓ using ridge regression [17]. To this end, let Φ denote all column-wise concatenated feature vectors $\phi(\mathrm{pa}(x_t))$ that are assigned to the leaf. Likewise, and in the same order, we concatenate all the desired predictions column-wise into a matrix U. The matrix A_ℓ now has the closed-form solution

$$A_\ell := U\Phi^\top \left(\Phi\Phi^\top + \gamma I\right)^{-1},$$

where $\gamma > 0$ is the ridge regularization parameter.

To determine the covariance matrix Σ_ℓ, we first use the ground truth to compute the residual vectors $r_t := x_t - A_\ell\phi(\mathrm{pa}(x_t))$. The set of all the residual vectors is then used to estimate a matrix $\hat{\Sigma}_\ell$ by means of the sample covariance. While the estimate of A_ℓ is generally quite accurate, we observed that the covariance estimate $\hat{\Sigma}_\ell$ may become inaccurate for high dimensions and small sample sizes. In some cases the estimated matrices are even singular. We therefore regularize our initial estimate by projecting $\hat{\Sigma}_\ell$ to an isotropic target with full rank,

$$\Sigma_\ell := d^{-1}\,\mathrm{tr}\left(\hat{\Sigma}_\ell\right) I,$$

a measure that proved to be important for the success of our approach. We also experimented with a convex combination of the sample covariance matrix and a diagonal shrinkage target [24], but it did not improve the results.

2.3.2 Training the Tree Structure

To determine the tree structure, we use a greedy training procedure, as is commonly used in the literature [6, 10]. We start with a single node and recursively split leaf nodes by selecting the best among a set of hyperplane splits. Each candidate split is sampled from a proposal distribution p_s, e.g., by uniformly sampling a data point and a coordinate. A candidate split s at leaf ℓ introduces child nodes u and v, each of which receives a subset of the data present at ℓ. After estimating a leaf model for u and v according to section 2.3.1, we determine the quality of the proposed split by measuring the resulting reduction in residual error,

$$Z_s := E_\ell - (E_u + E_v)\,,$$

where E_η is given as the sum of squared residuals norms,

$$E_\eta := \sum_{t\in\mathbf{D}_\eta} \|r_t\|^2.$$

The split that achieves the largest score Z_s is selected and the training proceeds recursively.

Implementation of DFMs is easy and analytical solutions for the score function and the least squares regressor make learning very efficient. The required training time can be controlled by the number of trees, their depth and the number of tested splits. Algorithm 1 summarizes DFM training.

3. Latent space view and comparison to Hidden Markov Models

In this section we will compare our Dynamic Forest Model with the class of Hidden Markov Models. For an easier comparison we formulate DFMs as a latent variable model, thereby making explicit the relationship between decision tree leaves and latent states.

Consider a tree $\mathcal{T}_c$ with l leaf nodes. By introducing latent variables z_t with l states we arrive at the latent variable model depicted in Figure 2c for order $K = 2$. In this formulation the joint distribution of an observed sequence X and latent variables Z is given by

Algorithm 1 Probabilistic DFM training

input: time-series data $\mathbf{D}$
input: number of ensemble trees $C > 1$
input: number of split tests $M \geq 1$
input: split proposal distribution p_s
output: dynamic forest $\{\mathcal{T}_c\}_{c=1,\ldots,C}$
procedure TRAINDFM($\mathbf{D}, C, M, p_s$)
  for $c = 1, \ldots, C$ **do**
    Bootstrap resample the training set
    $\mathcal{T}_c \leftarrow$ a growable root node ρ
    while there is a growable leaf ℓ in $\mathcal{T}_c$ **do**
      $Z^* \leftarrow -\infty$
      for $m = 1, \ldots, M$ **do**
        Sample split $s \sim p_s$
        Partition data at ℓ w.r.t. s
        $Z_s \leftarrow$ Compute score for s
        if $Z_s > Z^*$ **then**
          $(s^*, Z^*) \leftarrow (s, Z_s)$
        end if
      end for
      Split leaf ℓ using split s^*
    end while
    for leaf ℓ in tree $\mathcal{T}_c$ **do**
      Build least squares regressor at ℓ
    end for
  end for
  return Ensemble $\{\mathcal{T}_c\}_{c=1,\ldots,C}$
end procedure

$$p(X, Z) = \prod_{t=K+1}^{T} p(x_t \mid \mathrm{pa}(x_t))\, p(z_t \mid \mathrm{pa}(z_t)).$$

Since the latent states are deterministic predictions from observations, their distributions can be understood as delta functions $p(z_t \mid \mathrm{pa}(z_t)) = \delta(z_t - \ell(c, t))$.

In the DFM, information between latent variables has to flow through observed variables, whereas in HMMs the latent variables have a direct interaction. Because the latent states are determined from observations only they are conditionally independent given the observed sequence. This is different to Hidden Markov Models. To be precise the DFM encodes the following conditional independence statements for all $t > K + 1$:

$$z_t \perp\!\!\!\perp \{z_1, \ldots, z_{t-1}\} \mid \mathrm{pa}(z_t),$$
$$x_t \perp\!\!\!\perp \{x_1, \ldots, x_{t-(K+1)}\} \mid \mathrm{pa}(x_t).$$

It is this factorization that allows to compute the marginal likelihood in a single summation,

$$\begin{aligned} \log p(X) &= \log \sum_{z_{K+1}, \ldots, z_T} p(X, Z) \qquad (4) \\ &= \sum_{t=K+1}^{T} \log p(x_t \mid x_{t-K}, \ldots, x_{t-1}, \ell(c, t)). \end{aligned}$$

Under the assumption of balanced trees, the cost for the computation of log-likelihoods in a DFM is $\mathcal{O}(\log(l)\, T)$, which is sublinear in the number of latent states. The time complexity of Hidden Markov Models for the same task scales according to $\mathcal{O}(l^2 T)$, which is quadratic.

The additional efficiency in our model relies on two implicit assumptions: 1. We assume that we can identify the correct latent state. At time t, we put the entire probability mass on a single latent state that we select based on the feature vector $\phi(\mathrm{pa}(x_t))$. Our approach thus stands and falls with the design of this feature vector and the information it encodes. A Hidden Markov Model on the other hand can incorporate prior knowledge from the application domain (e.g., occlusion reasoning, object-object interactions, or compositionality [3]) by refining the model used for the hidden state sequence; 2. We assume that long-range dependencies are negligible. Whereas we do not model interactions beyond order K, Hidden Markov Models do have a long-term memory due to the Markov process on the hidden state sequence, thereby rendering the observation sequence non-Markov ([14], Section 1.3.3).

4. Experiments

We demonstrate the usefulness of DFMs on three different tasks: action recognition, motion completion and prediction of 3D motion from 2D inputs. Our experiments are based on the MSRC-12 dataset [12] and a modified version of the CMU dataset[2] as used by Wang *et al.* [29].

Technically, we represent a human motion sequence of length T as a temporal sequence of d-dimensional body poses. This yields a matrix $X \in \mathbb{R}^{d \times T}$ which can be used as part of an ensemble training according to section 2. Depending on the dataset, the individual poses in the columns of X are given in either joint angles or world coordinates. In our experiments, the feature mapping $\phi(\mathrm{pa}(x_t))$ concatenates all vectors in the subsequence $\mathrm{pa}(x_t)$ and adds a constant that models an intercept and allows for affine regression functions.

4.1. Human Action Recognition

The MSRC-12 dataset comprises sequences of people performing a total of 12 iconic and metaphoric gestures. Every sequence is the output of the Kinect's human body tracker, which gives a noisy estimate of 20 joints. These are defined in xyz-world coordinates, resulting in a $d = 60$ dimensional vector x_t per frame.

We use the iconic gestures from this dataset, which amounts to 296 sequences of about 1000 frames length each. The task is to classify sequences to their six iconic gesture classes

$$G = \{\text{Duck}, \text{Goggles}, \text{Shoot}, \ldots \\ \text{Throw}, \text{Change weapon}, \text{Kick}\}.$$

[2] Dataset obtained from `mocap.cs.cmu.edu`.

Table 1: Action classification results. **(a)** Accuracies and runtimes of DFMs and four baseline models. DFMs outperform all other evaluated methods. **(b)** Classification accuracies of DFMs as a function of depth, order, and number of trees. The result in bold is shown in more detail in **(a)**.

(a) Comparison to baselines.

Gesture	k-NN	SVM	HMM		DTW	DFM
				+PCA		(ours)
Duck	88.0	88.0	94.0	**98.0**	**98.0**	96.0
Goggles	70.0	84.0	54.0	70.0	**88.0**	**88.0**
Shoot	71.4	79.6	36.7	73.5	46.9	**85.7**
Throw	84.0	76.0	64.0	**90.0**	86.0	**90.0**
Change weapon	60.4	81.3	35.4	77.1	79.2	**87.5**
Kick	95.9	89.8	**98.0**	**98.0**	87.8	**98.0**
Accuracy	78.4	83.1	63.6	84.4	81.1	**90.9**
Runtime (sec./seq.)	9.24	189.23	0.50	0.45	107.55	**0.23**

(b) DFM results.

Trees	Depth	Markov order		
		1	2	3
1	4	69.6	69.6	62.5
12	1	86.5	88.2	87.8
	2	76.4	88.5	84.1
	3	87.8	90.2	89.2
	4	87.2	**90.9**	90.5

Each class comprises approximately 50 sequences coming from 30 different persons. In order to assess the generalization capabilities of our approach across persons, we employ 5-fold leave-person-out cross-validation, i.e., each fold consists of 24 persons for training and 6 persons for testing. We train $|G|$ different DFMs $\{\boldsymbol{\mathcal{T}}_g\}_{g\in G}$ per fold, one for each gesture, according to Algorithm 1. We report classification accuracies obtained with dynamic forests consisting of 12 trees, all of which were trained with 2^{10} tested splits. We use a fixed ridge regularization of 10^{-5} but vary the tree depth and Markov order.

In order to assign an unseen test sequence X to its corresponding class g^*, we compute the log-likelihood of the sequence under each of the individual per-gesture models and assign it to the class maximizing this quantity,

$$g^* := \operatorname*{argmax}_{g'\in G} \log p\left(X \mid \boldsymbol{\mathcal{T}}_{g'}\right), \tag{5}$$

where the log-likelihoods are given according to (4).

Baseline methods. We compare DFMs to four different baseline methods. Two of them, k-nearest neighbours (k-NN) and Support Vector Machines (SVM), are standard classification methods, the other two, Hidden Markov Models (HMM) and Dynamic Time Warping (DTW), are more specialized dynamic models and tailored to time-series data.

For the k-NN and SVM approaches we classify all length-four subsequences in a given test sequence. The class label of the entire sequence is determined by taking the majority vote over all of those subsequences. In order to classify a subsequence using k-NN we compare it to all $\approx 256{,}000$ subsequences of the training set and find the k-nearest neighbours with respect to the Euclidean distance. The parameter k is set to 6, which yields best performance on the test set in the range $k = 1, \ldots, 8$. The SVM classifier is also trained on all $\approx 256{,}000$ subsequences. We use the implementation [9] with a one-vs-rest classifier and a Gaussian RBF kernel.

DTW is a powerful time-series classifier that aligns two sequences by computing a possibly nonlinear warping path between them, thereby ignoring variations in duration and speed. We use the reference implementation of the FastDTW authors [23] and vary the search radius between 2^4 and 2^8. A test sequence is classified by calculating the warp distance to all training sequences (normalized by path length) and assigning it to the class of the training sequence with minimal warp distance.

For the HMM experiments we use the implementation of [21], training one HMM per gesture class. The original authors tuned their implementation for the same type of Kinect skeletal data we use. The implementation supports the use of raw features and PCA-reduced features (+PCA) and we report results for both options. The PCA features are obtained by constructing subsequences of length four and using the coefficients of the first 12 principal components. This is a powerful preprocessing step that is necessary for the HMM to work. To find the number of hidden states, we perform model selection, testing 5, 10, 20, and 40 hidden states, and report the best test performance. At test-time we classify a sequence by evaluating the marginal log-likelihood for each HMM and assign the sequence to the class of highest likelihood.

4.1.1 Results

Table 1a shows the quantitative results of our DFM and a comparison to the baseline methods. The worst performance was delivered by the k-NN approach (78.4%), which is not surprising given its naïve exhaustive search. Overall the time-series models tend to perform better than the simple classification baselines. That said, DTW (81.1%) still falls short of the SVM accuracy (83.1%), but mostly due to its weak performance on the "Shoot" class (46.9%). Varying the search radius does not alleviate this problem and the maximum is already attained for 2^4. For the HMM the best result is obtained with 10 hidden states. Note that the HMM requires strong preprocessing: Using raw data the performance is inferior at 63.6% and only the PCA features make the HMM perform at 84.4%. Our DFM achieves an accuracy of 90.9% and outperforms all other evaluated methods, notably without relying on any form of preprocessing. This indicates that DFMs are very robust with respect to the fea-

Table 2: Motion completion results. **(a)** Walking: Joint angle $\overline{\text{RMSE}}$ of our DFM approach and two baseline models. LI = Linear Interpolation; GPDM = Gaussian Process Dynamical Model. **(b)** Jump forward and golf swing: World coordinate $\overline{\text{RMSE}}$ of our DFM approach as a function of tree depth and Markov order. Note the decrease of the error with increasing tree depth and/or Markov order.

Seq. no.	LI	GPDM		DFM
		B-GPDM	2-MAP	(ours)
07-01	88.72	65.38	68.69	**28.94**
07-02	90.13	64.47	64.43	**33.87**
08-01	113.71	70.05	72.61	**30.47**
08-02	112.26	72.11	90.80	**28.59**
12-02	62.02	37.06	**26.60**	35.83
12-03	58.87	40.40	**23.16**	29.79
16-21	68.10	32.87	53.13	**24.00**
35-03	61.07	**12.15**	20.74	16.45
Mean	81.86	49.31	52.52	**28.49**

(a) **Joint angle representation:** $\overline{\text{RMSE}}$ for 8 walking sequences and different models. The best results are highlighted in bold.

Gesture	Depth	Order			
		1	2	3	4
Golf swing	1	10.93	7.44	7.06	6.89
	2	9.15	9.62	6.19	5.96
	3	4.82	5.16	5.79	4.35
	4	4.58	**4.17**	4.75	4.31
Forward jump	1	17.28	17.07	18.45	16.38
	2	39.59	12.01	11.70	10.61
	3	12.94	11.47	10.53	10.44
	4	11.08	11.27	11.41	**9.19**

(b) **World coordinate representation:** $\overline{\text{RMSE}}$ for two more complex gestures. Results are shown for different tree depths and Markov orders.

Table 3: 3D from 2D results. $\overline{\text{RMSE}}$ of predicted 3D trajectory given a 2D input. Deeper trees perform consistently better, while the effect of the Markov order varies.

2D Input	Depth	Markov order			
		1	2	3	4
Top	1	6.08	6.18	8.17	7.72
	2	3.55	3.53	3.44	3.71
	3	2.19	1.92	1.71	1.55
	4	1.87	1.37	1.24	**1.23**
Side	1	7.31	8.03	8.96	9.37
	2	4.11	4.23	4.62	5.30
	3	3.13	3.23	3.35	3.76
	4	**2.97**	3.19	3.39	3.65

tures used. In Table 1b we show DFM results for different tree depths and Markov orders. For comparison, we also include accuracies for a single tree of depth 4. The results of the DFM are both better and more stable, proving the point that our approach reduces overfitting and increases predictive power.

4.2. Motion Completion

In this experiment we apply the DFM to a motion completion task on walking sequences and compare our results to a recent Gaussian Process Dynamical Model (GPDM). Furthermore, we demonstrate the suitability of the presented framework for more complex actions and provide some general insights into training of DFMs. All of our motion completion experiments use the CMU motion capture database, a rich and noiseless data source of people performing different activities. All sequences are given by 56 joint angles and an additional 6 parameters governing the global translation and orientation.

Motion completion refers to the task of recovering consecutively missing frames from a motion sequence. More specifically, given a complete motion sequence

$$X = \left(x_{1:i}\ ,\ x_{(i+1):(i+j)}\ ,\ x_{(i+j+1):T}\right) \in \mathbb{R}^{d\times T},$$

we remove the subsequence $x_{(i+1):(i+j)}$ of length j in the middle and estimate $\hat{x}_{(i+1):(i+j)}$ from the remaining frames based on our model.

As in [29], we use the following three preprocessing steps for all CMU sequences: 1. A modified skeleton (reducing the number of degrees of freedom from 62 to 50); 2. Temporal downsampling by a factor of four; 3. Centering of all variables.

After that, we take the first $T = 50$ frames of the 8 test sequences listed in Table 2a and consider 12 different starting positions $i \in S = \{4, \dots, 15\}$ for the removal of $j = 31$ frames. The sequences come from 5 different subjects walking at different speeds and with different styles. Infilling of the missing frames involves training of a DFM $\{\mathcal{T}_c\}_{c=1,\dots,C}$ with a total of 29 sequences, all of them preprocessed in the same way as described above and none of them part of the test set. The estimate for missing frame $j' \in \{1, \dots, 31\}$ in run i is then given by the conditional expectation

$$\begin{aligned}\hat{x}^{(i)}_{i+j'} &= \mathbb{E}\left[x^{(i)}_{i+j'}\middle|\text{pa}\left(x^{(i)}_{i+j'}\right)\right] \\ &= \frac{1}{C}\sum_{c=1}^{C} A_{\ell(c,i+j')}\phi\left(\text{pa}\left(x^{(i)}_{i+j'}\right)\right).\end{aligned}$$

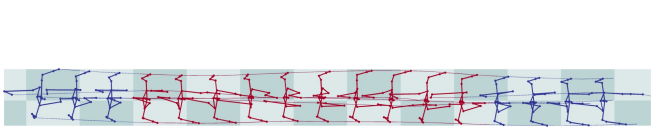
The estimates $\hat{x}^{(i)} = (\hat{x}^{(i)}_{i+j'})_{j'=1,\dots,j}$ of all 12 runs are subsequently combined to give an average RMS error per frame

$$\overline{\text{RMSE}}(\hat{X}) := \frac{1}{|S|}\sum_{i\in S}\sqrt{\sum_{j'=1}^{j}\frac{\|\hat{x}^{(i)}_{i+j'} - x_{i+j'}\|^2}{j}}$$

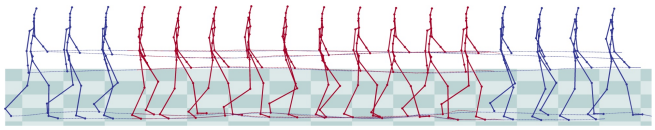
as a measure of reconstruction quality.

Although a direct comparison to the GPDM is illustrative, our model is not limited to simple actions such as walking. In fact, complex actions benefit more from our nonlinear and non-parametric approach. To fortify this claim, we train two additional DFMs on more challenging gestures: forward jump and golf swing. In particular, we use the same set of preprocessing steps as before and train DFMs on 7 (forward jump) and 8 (golf swing) training sequences, with 1 sequence in each category reserved for testing. Both test sequences are missing 31 frames and the RMSE is again averaged over 12 runs.

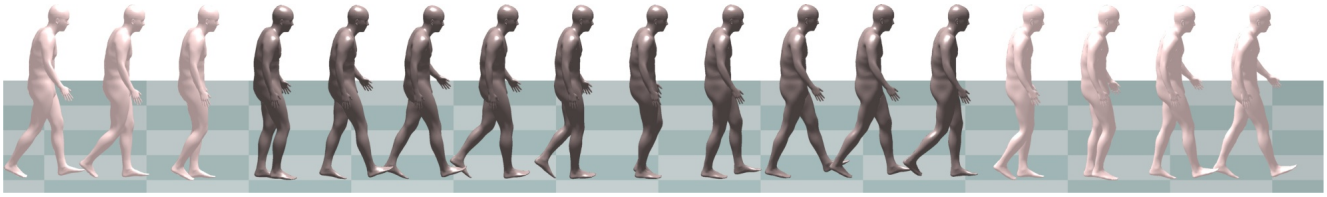
While our experiments on walking sequences use a representation in joint angles to allow comparison with the results in [29], our experience has shown that DFMs perform better when trained on a representation in xyz-world coordinates. In particular, the ability to benefit from deep trees



(a) Input: 2D trajectory (top view).



(b) Output: Stick model visualization of predicted 3D trajectory.



(c) Output: SCAPE model visualization of predicted 3D trajectory.

Figure 3: Visualization of a 2D→3D experiment. In **(a)**, we show every third frame of a 2D input sequence for which we want to reconstruct the red subsequence in 3D. In **(b)** and **(c)**, we show two animations of the output from our model: A stick model (top right) and a SCAPE model that was fitted based on our reconstruction (bottom). Ground truth data is shown in blue/light skin, reconstructions in red/dark skin.

and to map distinct parts of a motion sequence to distinct linear systems works well in the xyz-domain. We therefore convert the involved sequences to xyz-coordinates and use this new representation for our two additional motion completion experiments.

4.2.1 Results

Table 2a summarizes the quantitative results of the experiments on walking sequences and compares our findings with simple linear interpolation (LI) and the Gaussian Process Dynamical Model (GPDM). As expected, linear interpolation performs worst, with an average $\overline{\text{RMSE}}$ of 81.86. The original authors of the GPDM [29] consider four different learning procedures and we restate the numbers for the two best performing approaches: B-GPDM, which adds a balancing term to MAP estimation, and 2-MAP, which is a two-stage process involving a Hybrid Monte Carlo version of EM and Scaled Conjugate Gradient. The former performs slightly better (49.31 vs. 52.52). We compare these results with a DFM consisting of 12 trees. Our method achieves a lower $\overline{\text{RMSE}}$ on 5 out of 8 test sequences.

Table 2b shows our results for the actions 'forward jump' and 'golf swing'. DFMs are suited for these more complex actions just as well. Both gestures take full advantage of our distributed approach and the minimum error is reached for a tree depth of 4. The Table also suggests that depth is more important than order: While we do see some improvement with increasing Markov order, the error decreases much more substantially with increasing tree depth. Our assumption that long-range dependencies are only of minor importance thus seems to hold.

4.3. Predicting 3D from 2D

For our last demonstration we consider the task of predicting 3D motion from 2D inputs. In combination with the output of a pose estimation pipeline, we see potential applications like automatic character animation from videos.

In particular, we use the same walking sequences and preprocessing steps as we did in section 4.2 but train our model with pairs of 2D inputs and 3D outputs, where we obtain the 2D data from orthographic projections to the xz-plane (top view) and the yz-plane (side view).

Our results are summarized in Table 3. When using 2D views from the top, our findings are in accordance with those in the previous section, i.e., the $\overline{\text{RMSE}}$ decreases considerably when using deeper trees and higher Markov orders, although the former seems to have a higher influence. For example, note the decrease in error of 84% between the best performing model, a tree of depth 4 and Markov order 4, and its counterpart of depth 1. For 2D views from the side, the test sequences cannot take advantage of higher orders, but the benefit from deep trees still leads to a substantial improvement of 59% compared to trees of depth 1.

Finally, we want to complement our numerical assessment of the model quality with a visual inspection of the reconstructed sequences: Figure 3a shows a stickman visualization of the available 2D input data (projections to xz-plane). Based on the predicted 3D motion trajectories from our model (Figure 3b), we can then produce a realistic looking animation of a SCAPE model ([1], Figure 3c).

4.4. Computational aspects

One of the major benefits in using DFMs instead of latent variable models lies in their combination of being both flexible and tractable. The calculation of an exact log-likelihood for our action classification experiment takes only 0.04 sec. when using a dynamic forest with 12 trees of depth 4. We can thus classify a sequence in around 0.23 sec. Likewise, one synthesis step in our motion completion experiment takes 0.05 sec. All numbers refer to our Matlab

implementation.[3] Table 1a shows how that compares to the other methods. The DFM is almost twice as fast as an HMM and magnitudes faster than the remaining methods.

5. Conclusion

We have proposed the Dynamic Forest Model (DFM) as a new approach to human motion modelling, generalizing autoregressive trees and introducing new training and regularization techniques. Our presentation of DFMs in terms of latent variables has allowed a direct comparison to HMMs. Instead of relying on a latent space, we use a non-parametric and nonlinear Markov model that allows exact and efficient inference while at the same time being able to represent complex conditional mixture distributions. We proposed the use of bagging to avoid overfitting, which effectively complements other regularization techniques like ridge regression and isotropic covariance estimation.

The effectiveness of this approach has been demonstrated in three different application scenarios, namely action recognition, motion completion, and prediction of 3D trajectories from 2D inputs. The comparison with popular baselines and state-of-the-art latent variable models like HMMs and GPDMs has shown that DFMs perform excellent in those areas, while still being computationally efficient. Given the positive findings of this work, we believe that human motion models based on DFMs could be useful in many areas beyond the ones mentioned in this work, including tracking, character animation and pose correction.

Acknowledgements The first author was supported by Microsoft Research through its PhD Scholarship Programme.

References

[1] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis. SCAPE: Shape completion and animation of people. *SIGGRAPH*, 2005. 7

[2] D. Barber. *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 2012. 1

[3] M. Brand, N. Oliver, and A. Pentland. Coupled hidden Markov models for complex action recognition. *CVPR*, 1997. 4

[4] C. Bregler. Learning and recognizing human dynamic in video sequences. *CVPR*, 1997. 1

[5] L. Breiman. Bagging predictors. *Mach. Learn.*, 1996. 2

[6] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, 1984. 3

[7] O. Cappé, S. J. Godsill, and E. Moulines. An overview of existing methods and recent advances in sequential Monte Carlo. *Proceedings of the IEEE*, 2007. 1

[8] O. Cappé, E. Moulines, and T. Rydéen. *Inference in Hidden Markov Models*. Springer, 2005. 1

[9] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2011. 5

[10] A. Criminisi, J. Shotton, and E. Konukoglu. Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning. *Found. Trends. Comp. Graphics and Vision*, 2012. 3

[11] J. Fan and Q. Yao. *Nonlinear Time Series: Nonparametric and Parametric Methods*. Springer, 2005. 2

[12] S. Fothergill, H. M. Mentis, P. Kohli, and S. Nowozin. Instructing people for training gestural interactive systems. *CHI*, 2012. 4

[13] E. Fox, E. Sudderth, M. Jordan, and A. Willsky. Nonparametric Bayesian learning of switching linear dynamical systems. *NIPS*, 2008. 1

[14] A. M. Fraser. *Hidden Markov Models and Dynamical Systems*. SIAM, 2008. 1, 4

[15] N. Gordon, D. Salmond, and A. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proc. F. Radar and Signal Proc.*, 1993. 1

[16] G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 2002. 1

[17] A. Hoerl and R. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 1970. 3

[18] K. Ito and K. Xiong. Gaussian filters for nonlinear filtering problems. *IEEE Trans. on Automatic Control*, 2000. 1

[19] S. J. Julier and J. K. Uhlmann. A new extension of the Kalman filter to nonlinear systems. *Proc. AeroSense: 11th Int. Symp. Aerospace/Defense Sensing, Simulation and Controls*, 1997. 1

[20] C. Meek, D. M. Chickering, and D. Heckerman. Autoregressive tree models for time-series analysis. *SIAM*, 2002. 2

[21] S. Nowozin and J. Shotton. Action points: A representation for low-latency online human action recognition. Technical report, 2012. MSR-TR-2012-68. 5

[22] V. Pavlović, J. M. Rehg, and J. MacCormick. Learning switching linear models of human motion. *NIPS*, 2000. 1

[23] S. Salvador and P. Chan. Toward accurate dynamic time warping in linear time and space. *Intell. Data Anal.*, 2007. 5

[24] J. Schaefer and K. Strimmer. A shrinkage approach to large-scale covariance matrix estimation and implications for functional genomics. *Stat. Appl. Genet. Molec. Biol.*, 2005. 3

[25] L. Sigal, A. Balan, and M. J. Black. HumanEva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion. *IJCV*, 2010. 1

[26] G. W. Taylor, G. E. Hinton, and S. Roweis. Modeling human motion using binary latent variables. *NIPS*, 2007. 1

[27] R. Urtasun, D. J. Fleet, and P. Fua. 3D people tracking with Gaussian process dynamical models. *CVPR*, 2006. 1

[28] R. Urtasun, D. J. Fleet, and N. D. Lawrence. Modeling human locomotion with topologically constrained latent variable models. *2nd Workshop on Human Motion*, 2007. 1

[29] J. M. Wang, D. J. Fleet, and A. Hertzmann. Gaussian process dynamical models for human motion. *PAMI*, 2008. 1, 4, 6, 7

[30] J. M. Wang, D. J. Fleet, and A. Hertzmann. Optimizing walking controllers for uncertain inputs and environments. *ACM Trans. Graphics*, 2010. 1

[31] J. M. Wang, S. R. Hamner, S. L. Delp, and V. Koltun. Optimizing locomotion controllers using biologically-based actuators and objectives. *ACM Trans. Graphics*, 2012. 1

[3] http://ps.is.tue.mpg.de/person/lehrmann