Robust Generation of Dynamical Patterns in Human Motion by a Deep Belief Nets

Sainbayar Sukhbaatar SAINAA@SAT.T.U-TOKYO.AC.JP
Department of Mathematical Informatics,
Graduate School of Information Science and Technology, The University of Tokyo

Takaki Makino MAK@SAT.T.U-TOKYO.AC.JP
Collaborative Research Center for Innovative Mathematical Modelling,
Institute of Industrial Science, The University of Tokyo

Kazuyuki Aihara AIHARA@SAT.T.U-TOKYO.AC.JP
Institute of Industrial Science
The University of Tokyo

Takashi Chikayama CHIKAYAMA@LOGOS.IC.I.U-TOKYO.AC.JP
Department of Electrical Engineering and Information Systems,
Graduate School of Engineering, The University of Tokyo

Editors: Chun-Nan Hsu and Wee Sun Lee

Abstract

We propose a Deep Belief Net model for robust motion generation, which consists of two layers of Restricted Boltzmann Machines (RBMs). The lower layer has multiple RBMs for encoding real-valued spatial patterns of motion frames into compact representations. The upper layer has one conditional RBM for learning temporal constraints on transitions between those compact representations. This separation of spatial and temporal learning makes it possible to reproduce many attractive dynamical behaviors such as walking by a stable limit cycle, a gait transition by bifurcation, synchronization of limbs by phase-locking, and easy top-down control. We trained the model with human motion capture data and the results of motion generation are reported here.

Keywords: Deep Belief Net, Restricted Boltzmann Machine, motion generation, bifurcation, limit cycle, phase-locking

1. Introduction

Deep Belief Nets (DBNs) are deep generative models that have been successfully applied to various machine learning problems, such as hand-written digit recognition and object recognition (Hinton and Salakhutdinov, 2006). The deep architecture is achieved by stacking up Restricted Boltzmann Machines (RBMs), two-layer stochastic binary neural networks. Their unsupervised learning and hierarchical structure made DBNs a promising approach in computer vision.

Our goal is to utilize the multi-layer deep structure of DBNs in generation of human motion, a high dimensional temporal pattern. We are interested in robust and flexible motion generation, such as walking by a stable limit cycle, a gait transition by bifurcation, phase-locking between limb motions, and easy control by a simple top-down signal. Those

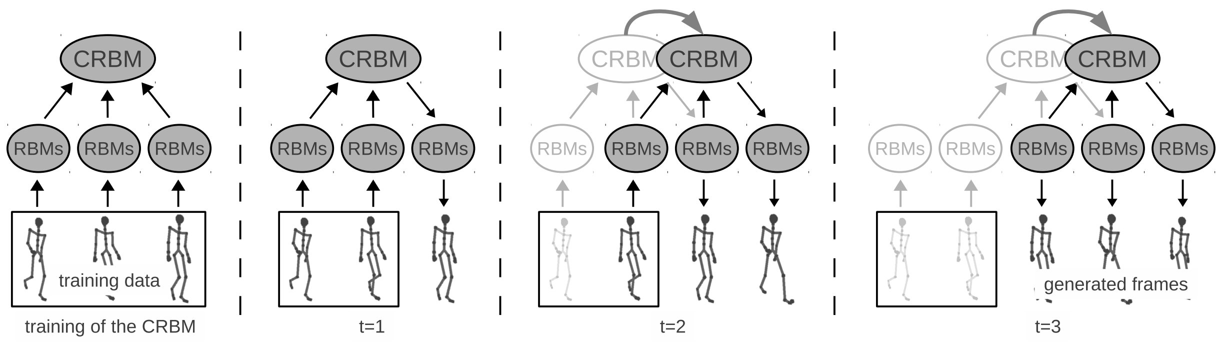


Figure 1: Motion generation steps of Temporal Deep Belief Nets

properties are inspired by models of Central Pattern Generators, a biological neural network for motion generation. Most CPG-based models generate motions by phase-locked mathematical oscillators. However, it is hard to generate high-dimensional motions, such as human motions, by those hand-crafted models.

Applications of DBNs has been restricted to static patterns. There are two obvious difficulties in generating dynamic patterns with DBNs. First, it is not clear how to utilize the hierarchical architecture of DBNs in temporal patterns. Second, RBMs, which are the building block of DBNs, are not capable of learning temporal constraints.

A recent work by Taylor et al. (2007) showed that RBMs can be modified to learn motion data. They added additional layers, which keep the past states of the visible layer, to an RBM so that it can learn the constraints between adjacent frames of motion. This model is called Conditional RBM (CRBM) because the sampling in the RBM is conditioned on those new layers. They also showed that two different motions can be learned by a single CRBM, and addition of little noise to the hidden layer can induce a stochastic smooth transition between the two motions. However, CRBMs have several shortcomings when it is used for our purpose. First, motion transitions in CRBMs are stochastic and hard to control. Second, training of CRBMs is unstable when the visible layer represents real valued motion data. Third, in order to generate motion, CRBMs have to keep several frames of actual motion.

Our key idea is to introduce a hierarchical structure into CRBM for robust motion generation. Our model, which we call a Temporal Deep Belief Net, consists of multiple RBMs and one CRBM. The RBMs work in parallel at the bottom of the CRBM. They are trained to encode motion frames into compact representations. It is also possible to decode motion frames back from compact representations using those RBMs. The CRBM, which sits on the top of the RBMs, is trained with sequences of compact representations encoded by the RBMs from training motions. During motion generation, the CRBM generates a sequence of compact representations, which are then transformed into motion frames by the RBMs (see Figure 1).

The main difference between our model and Taylor et al. (2007) is that ours has an additional layer of RBMs at the bottom. Therefore, the CRBM in our model learns transition constraints between compact representations, rather than raw motion frames. Also, previous motion frames are unnecessary for generating the next frame. This makes the learning so easier that the hidden layer of the CRBM is not necessary for learning a single motion. It is also possible to use this layer as a control layer when generating multiple motions, because sampling in the CRBM is conditioned on the layer. The control layer represents more abstract motion signals such as walking or running. A series of experiments show that the model can generate various motions in a robust and flexible way.

The rest of the paper is organized as follows. In Section 2, we review RBMs and DBNs. In Section 3, we propose Temporal Deep Belief Nets. In Section 4, the results from the experiments will be presented. In Section 5, we discuss the biological plausibility of our model and its relationship to Central Pattern Generator models. Finally, Section 6 concludes this paper.

2. Background

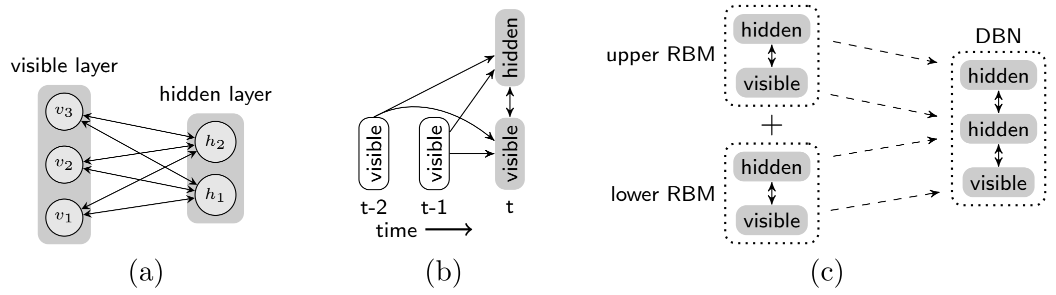


Figure 2: (a) A RBM (b) A CRBM for temporal patterns. (c) A DBN built from two RBMs.

2.1. Restricted Boltzmann Machine

Restricted Boltzmann Machines (RBMs) are a special type of Boltzmann Machines (Hinton, 2007) where units are separated into visible and hidden layers, and edges are restricted only between visible and hidden units (see Figure 2.a). The energy of an RBM is given as

$$E(\mathbf{v},\mathbf{h}) = -\sum_{i,j} v_i h_j w_{ij} - \sum_i v_i b_i - \sum_j h_j c_j, \tag{1}$$

where $\mathbf{v} = \{v_1, v_2, ..., v_n\}$ and $\mathbf{h} = \{h_1, h_2, ..., h_m\}$ are the state vectors of the visible and hidden layers. Units v_i and h_j have binary values of $\{0, 1\}$, and w_{ij} is the weight of the edge connecting v_i with h_j, and b_i and c_j are the biases of those units.

The stochastic behavior of the RBM is driven by the Boltzmann distribution

$$P(\mathbf{v},\mathbf{h}) \propto e^{-E(\mathbf{v},\mathbf{h})}, \tag{2}$$

which states that low energy states have high probability in the RBM. From (1) and (2), we can calculate the probability of i'th visible unit being 1:

$$P(v_i = 1|\mathbf{h}) = \frac{1}{1 + e^{-\sum_j w_{ij}h_j - b_i}}, \tag{3}$$

which is the sigmoid function of the total input. The same calculation can be made for hidden units. Therefore, one can get an unbiased sampling of the visible state $\mathbf{v}$ given the hidden state $\mathbf{h}$ using (3), and vice versa. With a few steps of Gibbs sampling (alternatingly sampling each layer), an approximate unbiased sampling of the whole network can be achieved. A gradient descent algorithm based on this approximation (called Contrastive Divergence) makes it possible to train RBMs efficiently (Hinton, 2002).

RBMs can handle real values with a little modification to the energy function. For example, to make visible units real-valued, we change the energy function to

$$E(\mathbf{v}, \mathbf{h}) = \frac{1}{2}\sum_i v_i^2 - \sum_{i,j} v_i h_j w_{ij} - \sum_i v_i b_i - \sum_j h_j c_j, \tag{4}$$

which includes the sum of the squares of the visible units to prevent them growing too large. Under the assumption that the visible units have $\mathcal{N}(0, 1)$ distribution, the value of i'th visible unit can be sampled from

$$\mathcal{N}(\sum_j w_{ij}h_j + b_i, 1), \tag{5}$$

which is a Gaussian distribution with the unit variance and the mean equal to the total input.

2.2. Conditional RBM

Conditional RBMs (CRBMs), first introduced in Taylor et al. (2007), are an extension of RBMs that can be trained on temporal patterns. A CRBM introduce additional layers to hold the past states of the visible layer (see Figure 2.b), which we call past-visible layers in this paper. There can be arbitrary number of past-visible layers. The past-visible units are always fixed the previous values of the visible units. Therefore, their values are not updated during Gibbs sampling. This one-way information flow is represented by directed edges. CRBMs can be trained with the same training algorithm as RBMs because the inputs from the past-visible units can be handled in the same way as biases. During training, the directed edges learn the temporal constraints between consecutive visible states.

To generate a motion by a CRBM, the visible and the past-visible units have to be real-valued because motions are described as real-valued joint-angles. Therefore, the edges connecting the visible and the past-visible layers have real-valued units on its both ends and this makes the weight update unstable.

2.3. Deep Belief Nets

The representative power of RBMs is limited due to their restricted connections. Deep Belief Nets (DBNs) solve this problem by stacking up several RBMs, where the hidden layer of an RBM becomes the visible layer of another RBM (see Figure 2.c). Even though DBNs have hierarchical structure with higher representative power, they can be easily trained by greedy layer-by-layer training of each RBMs (Hinton et al., 2006). In the case of the DBN in Figure 2.c, training has two steps. First, the lower RBM is trained with training data, ignoring the upper RBM. Second, the upper RBM is trained with compact representations encoded from the training data by the lower RBM. In other words, the lower RBM learns to encode patterns in the training data to compact representations, and the upper RBM learns to further encode patterns in those compact representations into more compact representations at its hidden layer.

When applied to computer vision, the lower RBM learns many simple features of training images, while the upper RBM learns to combine those simple features to construct more complex features. Such hierarchical features are useful in image processing and it is similar to the feature recognition in the visual cortex (Hubel et al., 1988). By using max-pooling and convolutional connections in a DBN, a very powerful image recognition system can be built (Lee et al., 2009). Since each RBM in a DBN encodes its visible state into compact representations at its hidden layer, DBNs can be also used as a dimension reduction method where training data is encoded in several steps to more and more compact representations.

3. Temporal Deep Belief Nets for Motion Generation

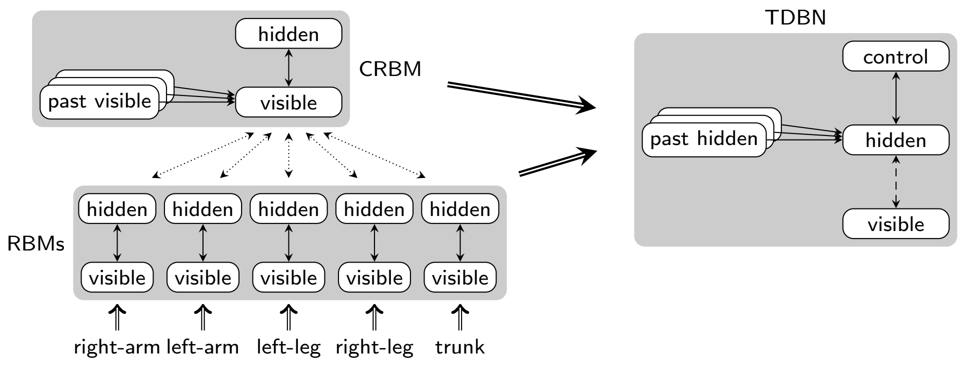


Figure 3: A TDBN is built by merging the hidden layers of RBMs with the visible layer of a CRBM

In this paper we propose a two-layer DBN model for motion generation, which we call "Temporal DBN" (TDBN). The lower-layer has multiple RBMs in parallel, instead of one RBM. This distributed architecture makes TDBNs robust and flexible. The hidden layers of those RBMs represent compact representations of motions frames. Those compact representations are used in the upper-layer, which has one CRBM to coordinate the total temporal behavior (see Figure 3).

Each RBM in the lower-layer correspond to a different body part. For example, the RBM corresponding to the right leg is only trained with the motion of the right leg. This division of training data is based on the assumption that joint-angles from the same limb are more correlated than two joint-angles from different limbs. We can expect more compact representations at the hidden layers by putting more correlated variables in the same RBM.

Using multiple RBMs in parallel instead of a single RBM significantly reduces the the number of learning parameters and, therefore, help us to overcome the curse of dimensionality. Each RBM have to learn only 7 to 21 dimensional data, instead of 56 dimensional full motion data. Actually, these RBMs can be viewed as one large RBM with sparse connections. However, disadvantage of using separate RBMs is the difficulty of generating coordinated whole body motions, such as walking and running. Also, the RBMs does not have temporal elements. Therefore, the lower-layer is only for learning static features of motion frames, leaving dynamic features to be learned by the upper-layer.

Above those RBMs, there is only one CRBM. The role of this CRBM is to connect the separate RBMs together and keep them in synchrony. Also, it learns dynamic features of the human motion through its directed edges. All of the hidden units of the RBMs in the lower-layer belong to the visible layer of the CRBM in the upper-layer. In other words, the CRBM is trained by compact representations of motion frames, rather than raw motions. Therefore, the visible units of the CRBM are no longer have to be real-valued. This significantly reduces the burden of the CRBM.

In TDBNs, a simple walking motion can be generated even if there was no hidden units in the CRBM. However, the hidden layer is useful because the sampling in the CRBM is conditioned on it. We can use this property to control the behavior of the CRBM, and we call the hidden layer of the CRBM as "control layer". We can associate different motions with different static patterns at the control layer. Such association can be made by setting the value of the control layer to the static pattern corresponding to the motion being learned, during positive weight updates. After this training, we can choose which motion to generate by simply setting the control layer to the corresponding pattern.

Training of the TDBN is straightforward. First, each RBM is trained with corresponding parts of motion frames. After the training, the RBMs learn to encode motion frame into compact representation at their hidden layers. Then, those compact representations are used as training data in training of the CRBM. Motion generation also has two steps. First, the CRBM generates a sequence of compact representations. Then, each hidden representations is converted to motion frame by the RBMs.

4. Experiments & Results

We first show experiments with the training by a simple walking motion to demonstrate the stability and phase-locking in a TDBN. After that, we show the result of experiments, in which a TDBN is trained with multiple motions to explore transitions among them. All motion capture data used in the following experiments are obtained from CMU dataset (http://mocap.cs.cmu.edu/). Each motion is represented by a sequence of 59 joint-angles at 40fps (down-sampled from the original frame rate of 120fps). Coordinate variables are omitted because they are not part of the motion generation.

We used five separate RBMs in the lower-layer, each corresponding to one limb or the trunk. The total number of their visible units is 59, same as the number of joint-angles. The visible layer corresponding to the trunk has 21 real-valued units, the largest among the RBMs. Arms and legs each have 12 and 7 joint-angles respectively. The hidden layers,

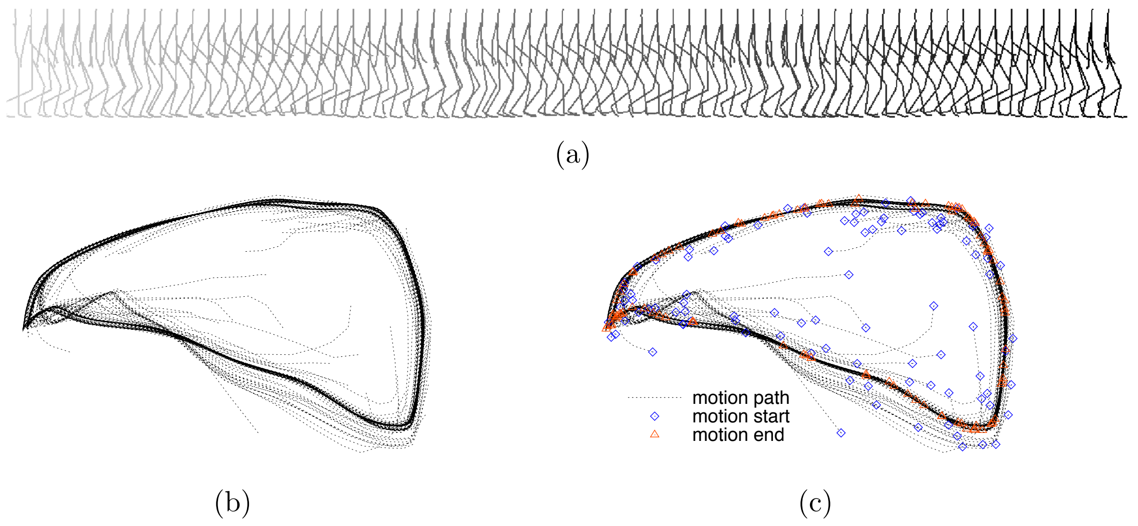


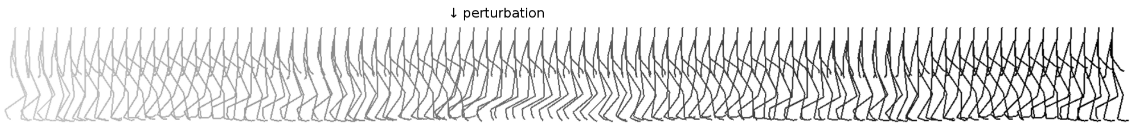
Figure 4: (a) A walking motion frames generated from the TDBN. (b) A limit cycle pattern emerges when 100 short walking motions are drawn as trajectories in a plane. We added Gaussian noise to the initial hidden state of each motion. (c) The starting and ending points of those trajectories

on other hand, each have 30 binary units. Therefore, the visible layer of the CRBM has 150 binary units. Because we keep the states of previous three time steps, the CRBM has three past-visible layers (or the past-hidden layers of the TDBN), which have 450 binary units in total. However, the control layer has relatively few units, 20 units in experiments with two motions and 30 units in experiments with three motions. In the experiments with multiple motions, the control layer will be clamped to static patterns. We used the contrastive divergence method for all training.

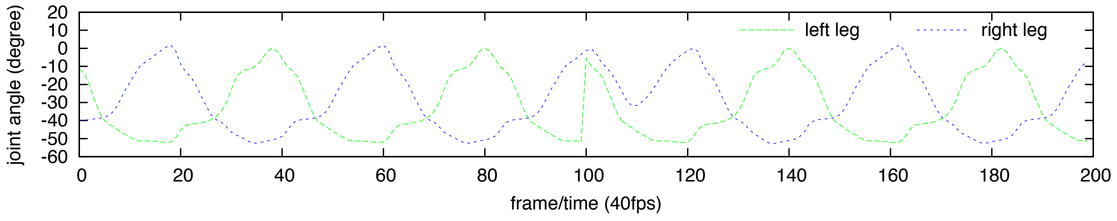
4.1. Stable Walking by a Limit Cycle

In this experiment, we trained the TDBN with a short walking motion to examine its stability. Here we used the word "stability" in the context of dynamic system, not necessarily the physical stability. First, the RBMs are trained by walking frames. Motion frames are learned in random order to prevent biases. After the training, the RBMs can encode real-valued joint-angles to a compact representation of binary units in a stochastic way. But, we used expected values during motion generation to reduce the noise induced by stochastic sampling.

Next, the sequence of compact representations, encoded from the walking motion by the RBMs, is used as training data in the training of the CRBM. The CRBM learns temporal constraints between compact representations by its directed edges. To start motion generation, the past-hidden layers (the past-visible layers in the CRBM) have to be set to compact representations sampled from the training sequence. Once it is done, the next compact representation is sampled by three Gibbs sampling in the CRBM, conditioned on those past states. By passing the current state of the hidden layer to the past-hidden



(a)



(b)

Figure 5: (a) A motion naturally returns to a normal walking motion after a perturbation. The perturbation is simulated by forcibly shifting the phase of the left leg by a half cycle. (b) Hip joint-angles from each leg are plotted. When the left leg is perturbed at frame 100, the right leg motion is smoothly synchronizing eventually with the left leg motion.

layers, subsequent compact representations can be generated. Finally, this sequence of compact representations is decoded back to motion frames by the RBMs. A walking motion generated in this way is presented in Figure 4.a.

Limit cycle behavior

Let us investigate the model as a dynamical system. Repetitive motions, such as walking, can be considered as limit cycles in the state space and the system state must continuously go around such limit cycle to generate a walking motion. The purpose of setting the past-hidden layers to appropriate values (sampled values from a original motion) in the start is to make sure that the system's initial state is on the walking limit cycle. However, if the CRBM succeeded in producing a stable limit cycle, any motion started from perturbed hidden states should eventually converge to the walking motion. To test this hypothesis, we generated 100 independent short walking motions by the TDBN with Gaussian noises added to the past-hidden units in the start of each motion to simulate perturbations. The result is shown in Figure 4.b where each motion is represented by a dotted trajectory. We used Principal Component Analysis (PCA) to reduce the dimension of motion data to a 2-dimensional plane. Initial and final states of each motion are marked in Figure 4.c for clarity. We can see that the motions started from perturbed hidden states are all ended up on a single dark loop, which represents the normal walking motion.

We have to clarify that our model does not take account of the physical interaction with the environment, so the dynamical stability in the CRBM does not mean physical stability in humanoid robot. To deal with physical stability, we would need to extend TDBN so that it can be used in motion generation with sensory input.

Synchronization of legs

We conducted another set of experiments to show the usefulness of the limit cycle behavior with a more practical example, where a perturbed motion returns to a normal motion. Generally, it is hard to generate perturbed motions by the RBMs, because such motions must have high energy since there are no perturbed motions in the training data. Fortunately, in our model, motion of each limb is generated by a separate RBM. Therefore, we can easily generate a perturbed motion by breaking the synchrony of those RBMs.

We simulated a perturbation in a walking motion by forcibly shifting the phase of the left leg. This is done by setting the hidden units of the left-leg-RBM to the values recorded from a previous normal walking motion with a slight delay for a limited period. The other RBMs are sampled with normal Gibbs sampling during that time. In other words, only the motion of the left leg is replayed from a previously recorded walking motion for short duration. As the result, the joint-angles of the left leg will have a discontinued jump at the start of the replay, which generates unnatural poses (see Figure 5.a). In Figure 5.b, the hip joint-angles from the both legs are plotted. The green line (representing the left leg) shifted its phase at frame 100 due to the forced perturbation. At the same time, the blue line (representing the right leg) lost its synchrony with the left leg. However, those two motions are coupled in the CRBM through their past states and such an out-of-sync motion must have high energy (i.e. low probability). One might ask why doesn't the right leg instantly change its phase to synchronize with the left leg. That is because such sudden movements also have low probability in the CRBM, since the training data did not include such movements. As the result, the motion of the right leg gradually synchronizes with the left leg and the motion returns to the walking limit cycle.

4.2. Gait Transition by Bifurcation

In the second set of experiments, we train the model on two different gait styles: walking and running. The goal of this experiment is two-fold: (1) to show that it is possible to learn and generate two different motions by a single TDBN, and (2) to demonstrate that the model can generate a natural gait transition motion. An advantage of having a single model for all motions is that transitions between multiple motions can be represented by the model, which would be probably impossible if we had separate models for each motions.

The control layer is used to control which gait should be generated by the model. Each unit in this layer will have the same fixed value: 0 for walking and 1 for running. We will call this value a *gait parameter*. The control layer will always be clamped except for the sampling from a reconstruction during training. Two separate motion capture data is used as training data, and there was no gait transition motion.

The training of the RBMs is the same as the previous experiments, but walking and running frames have to be learned in a random order to prevent a bias to one gait. The training of the CRBM is slightly easier than before because we can calculate $\langle v_i h_j \rangle_{data}$ directly because h_j is fixed to the gait parameter. During the reconstruction step, however, we have to unclamp the control layer to calculate $\langle v_i h_j \rangle_{recon}$.

After the training, we successfully generated separate walking and running motions by setting the gait parameter to 0 and 1 respectively. However, when we changed this gait parameter during a single motion generation, the model generated a gait transition motion.

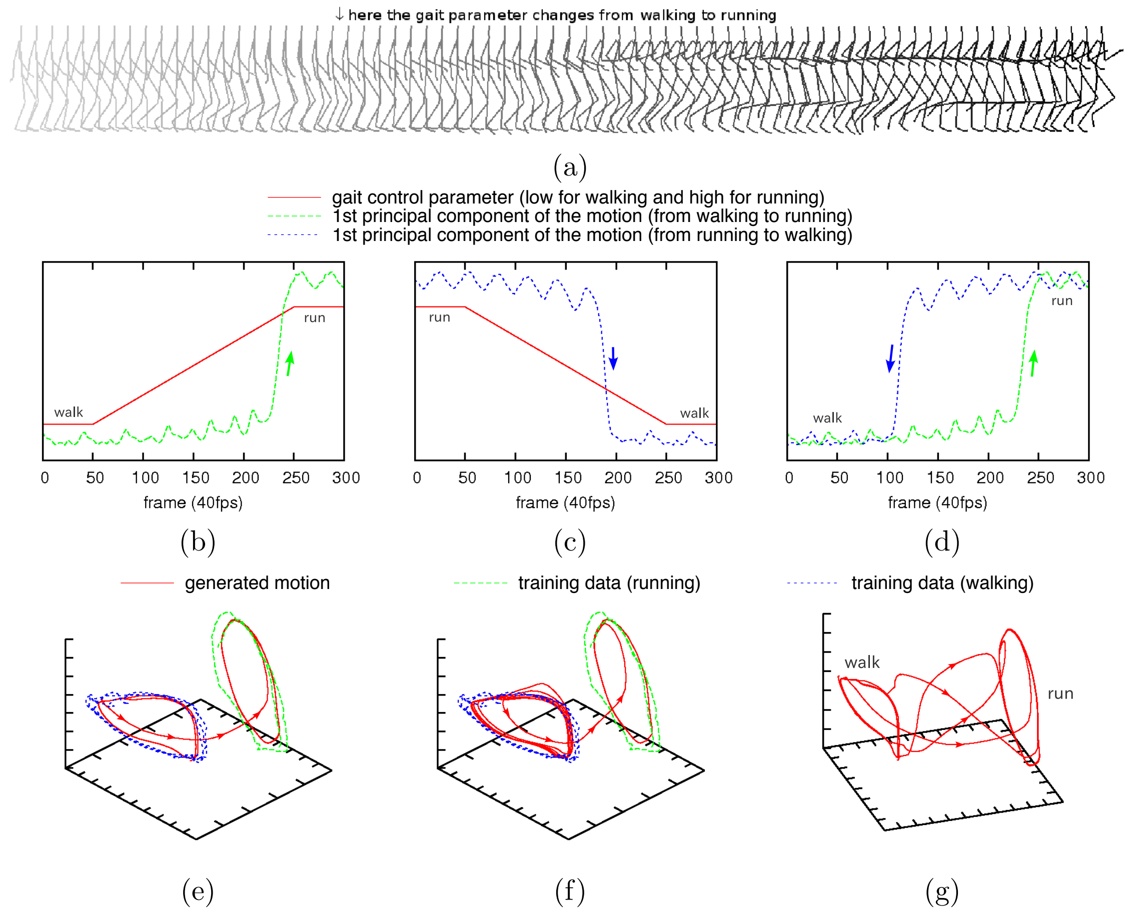


Figure 6: (a) A gait transition motion generated by the model. (b),(c) The first principal component of gait transition motions are plotted with the gait parameter value. (d) A hysteresis emerges when two transitions in (b),(c) are plotted to have the same gait parameters. (e) A bifurcation in the state space when the control parameter changes abruptly from walking to running. Two distinct limit cycles correspond to walking and running (first 3 principal components are plotted). (f) Even the control parameter changes slowly, the bifurcation occurs in the same way. (c) Bifurcation between two LCs is stochastic, thus the system travels different paths in each gait transition.

In Figure 6.a, we showed one such example where the gait parameter was changed from 0 to 1. From that moment, the motion smoothly transferred from walking to running in less than a half second.

We observed a smooth gait transition even when the control parameter changes abruptly from walking to running. One possible explanation for this smoothness is that the directed connections in the CRBM learned the continuity constraints of human motion from the training data. If it is true, even when top-down control changes abruptly, the lower level will try to generate a pattern that meets the continuity constraints as well as the top-down control constraint.

The number of units in the control layer is set to 20 in the experiments. More units will boost the effect of the gait parameter on the hidden layer, making the transition more sudden and short. On the contrary, less units will increase the effect of the past hidden states on the hidden layer, making transition slower and even impossible in some cases. Actually, we can replace those 20 units with a single unit with a 20 times large learning rate.

Emergence of hysteresis

We did another set of experiments where the gait parameter is gradually changed in 5 seconds from walking to running (see Figure 6.b). Also, the same experiment is done in the reverse direction: from running to walking (Figure 6.c). One would expect the gait transitions to occur at the same gait parameter threshold. However, if we overlap the two transitions to have the same gait parameter value in Figure 6.d, we can see that the threshold for a gait transition is dependent on the direction. Such direction dependent behaviors of dynamic systems is called *hysteresis*.

We are interested in this phenomenon because hysteresis is also observed in humans during gait transitions. Diedrich and Warren Jr (1995) explained hysteresis in the context of dynamical systems where walking and running motions are viewed as separate limit cycles (LCs). When the locomotion speed is slow, a walking LC is stable and a running LC is unstable. As the speed increases, the running LC also becomes stable. However, since the walking LC is still stable, the system will remain in the walking LC. Further increase in the speed will make the walking LC unstable and then the system will transfer to the stable running LC. Such a qualitative change in a dynamic system is called *bifurcation*.

The same explanation can be applied to our model. In the walking experiment, we showed that there is a stable walking LC in the hidden state space, but in this experiment, we trained the model on two different motions, so there must be two separate LCs. A gait transition happened when we changed the gait parameter. Therefore, it can be said that the gait parameter is controlling the stability of these LCs. In Figure 6.e, we showed a bifurcation from a walking LC to a running LC when the gait parameter changed suddenly from 0 to 1. In Figure 6.f, however, the gait parameter changed slowly. Interestingly, the system state keeps circling the walking LC until the gait parameter almost reaches to 1. Then suddenly, the system state leaves the walking LC and transfers to the running LC without losing its continuity (there are actually 5-10 frames in the transition path). Such a bifurcation between two limit cycles is called a saddle node bifurcation. In the saddle node bifurcation, there is a region where both attractors are stable and this produces a hysteresis in the system.

Figure 6.g demonstrates one of the advantages of our model: a stochastic behavior. Even though four motions of a gait transition are generated in the same condition, each motion path differs from one another because of the inherent stochastic behavior of Boltzmann Machines. Such variety of motions is important in some applications such as computer games where repeated identical motions would give an artificial look to characters.

4.3. Nonrhythmic Motion and Interruption

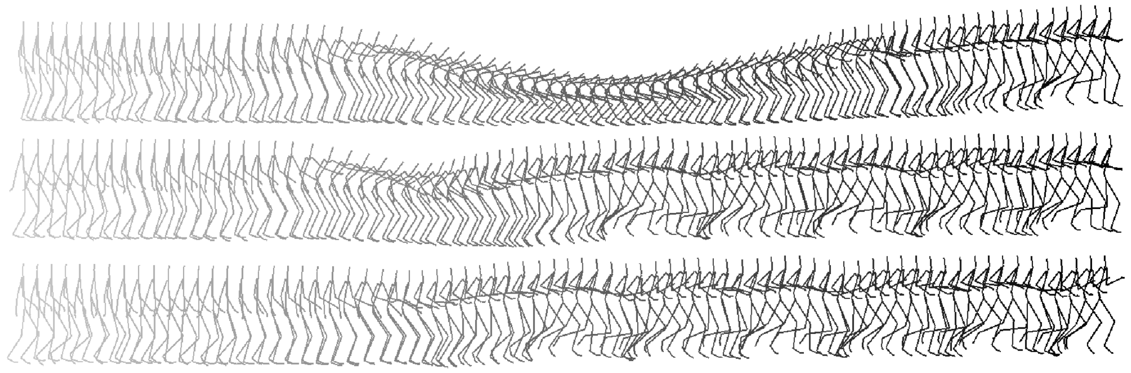


Figure 7: Sequential generation of three motions: walking, bending over, and running. (top) Running motion is started after the completion of the bending motion. (middle) The bending motion is interrupted by the running motion. (bottom) The interruption starts earlier.

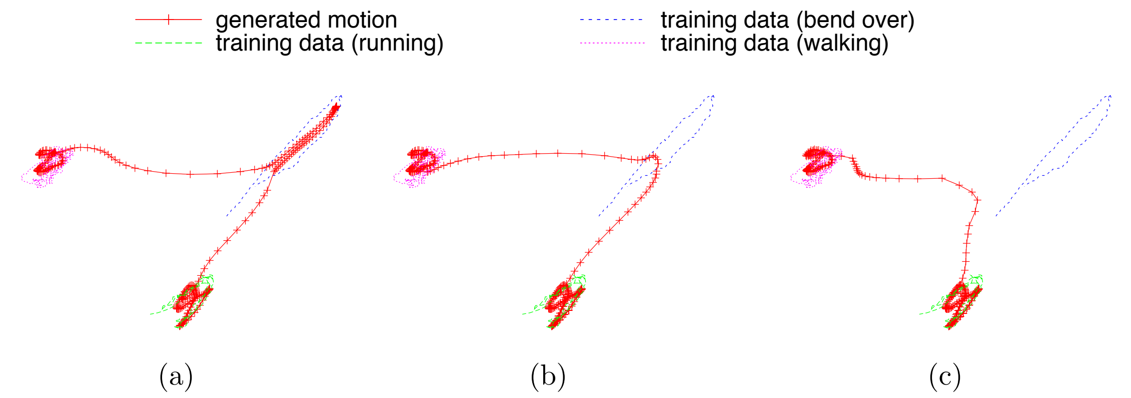


Figure 8: The fist two principal components of the motions in Figure 7 are plotted along with the original training motions. The motion starts from the walking region (green) and smoothly transfers to the bending region (blue) and ends at the running region (pink). In (b) and (c), however, the bending motion is interrupted before its completion.

Motions that can be learned by the model is not restricted to rhythmic motions. However, motions with too many still frames are difficult to learn because the model predicts the next frame based on only past three frames. In this experiment, we added a new motion of "bend over, scoop up and rise" to the training data. Since there are three motions to learn, we will divide the control layer into three equal parts each representing one motion. To generate a specific motion, all we have to do is to set the control units in the corresponding part to 1 and the other control units to 0. The learning algorithm remains the same as the previous experiments.

In Figure 7, the model generated a bending motion between walking and running motions. This is done by changing the control units from walking to bending, then from bending to running. We generated three versions of this motion to show the flexibility of the model. In the top figure, the bending motion is completed before being transformed into the running. This can be seen from the motion trajectory in Figure 8.a where each training motion is also plotted. However, if we set the control layer to running before the bending motion completes, the resulting motion will look like the middle figure where the bending is interrupted and naturally transferred into the running motion. If we see the corresponding motion trajectory in Figure 8.b, we understand that the motion actually reaches the bending region before turning into the running region without executing the bending motion. If we change the control to running earlier, the motion trajectory will change the direction before reaching the bending region (see Figure 8.c). A corresponding motion is shown at the bottom of Figure 7 where the bending motion is barely observable. Such flexibility is convenient in on-line applications such as computer games and robot control. The transitions shown in Figures 8.b and 8.c are difficult to generate by traditional methods where all possible transitions between two motions are synthesized in advance.

Unlike the walking and running motions, the bending motion has a fixed-point attractor rather than a limit cycle because it is not cyclic. The result from this experiment shows that a bifurcation between a limit cycle and a fixed-point attractor is possible in the model.

5. Discussions

5.1. A DBN as a biologically plausible model

Lee et al. (2009) proposed a convolutional DBN model for object recognition and they showed that their model can make use of unlabeled data to make useful features. Interestingly, the learned features had a hierarchical structure, where units in higher layers responded to whole or part of objects, while units in lower layers responded to lower level features, such as edges and points. This resembles to the hierarchy of the visual cortex, where neurons in early stages of the visual processing respond to small-spots or edges, while neurons in later stages spike when we see certain objects, such as human faces.

There is another connection between brains and DBNs, which is the similarity of their learning algorithms. Though we know little about how brains work, there are many simplified learning rules of the synaptic plasticity, such as Hebbian learning, Spike Timing Dependent Plasticity, and so forth. Basically, when two neurons spike simultaneously, the synapse connecting them become stronger or weaker depending on the spike timing and other external factors. Similarly in a DBN, when two units are ON, the weight between them increase if the values are sampled from the training data, and decrease if the values are sampled during reconstruction. This resemblance makes the DBN a good candidate when building a biologically inspired learning model.

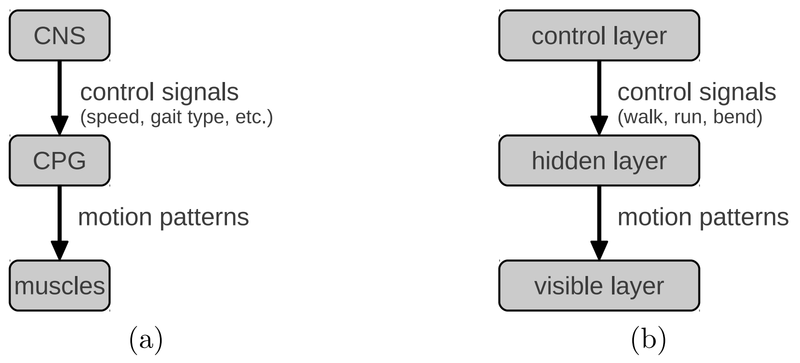


Figure 9: (a) The hierarchical structure of (a) the biological motion generation and (b) TDBNs

5.2. TDBN as Central Pattern Generator

Central Pattern Generators (CPGs), which are neural networks found in the spinal cord, have a crucial role in the biological motion generation. While CPGs can generate rhythmic motions autonomously, its behavior (e.g. speed, gait type) can be controlled by simple signals from the Central Nervous System (CNS) (Grillner, 1985; Shik, 1966).

There are many inspiring papers that tried to build an artificial CPG for motion generation (Ijspeert, 2008). Most of them studied CPGs in an abstract level using distributed mathematical oscillators (Collins and Stewart, 1992; Schöner et al., 1990). Phase-locking between those oscillators produce complex motion patterns similar to the motion of real animals (Ijspeert et al., 2007). While such motion controls are robust and manageable, they are often restricted to low dimensional motions because of the complexity limitation of carefully hand-crafted models.

The TDBN proposed in this paper can be considered as one implementation of CPG, because there are several properties of the TDBN that are similar to CPG-based models and biological CPGs. First, we demonstrated that the TDBN generates rhythmic motions by stable limit cycles, and gait transition motions by bifurcations. Second, when we consider that there are separate oscillators for different limbs, the TDBN successfully learned the phase-locking between those oscillators. Last, there is also a structural similarity between TDBNs and CPGs (see Figure 9). CPGs are capable of autonomously generating temporal patterns, while their behaviors can be controlled by simple top-down signals from the CNS. Our model achieved a similar structure. The hidden layer (and the past-hidden layers) can generate temporal patterns by itself, but a change in the control layer will affect the hidden layer, altering motion patterns (e.g from walking to running). In this paper, we generated 59 dimensional human motions, which shows that TDBNs are scalable to high dimensions.

5.3. Higher-order TDBNs

Our results show that TDBNs are a promising model for learning temporal patterns. The hierarchical structure of TDBNs makes it possible to control motion in an abstract level (i.e. walk, run, etc.) using the highest layer, while lower layers generate corresponding motion in a smooth way. If we train another TDBN on those abstract level patterns (i.e. the patterns of the control layer), we can generate a long-term abstract motion plan (e.g. walk-run-stop-bend) using the new TDBN. Then the lower TDBN would generate actual motions according to this plan. Such hierarchical temporal pattern generation will be useful for other applications too, such as speech generation.

6. Conclusion

In this paper, we introduced TDBNs, which are capable of learning and generating human motions. TDBNs have multiple RBMs at the bottom and one CRBM at the top. The RBMs encode still frames into compact representations, and the CRBM learns the temporal constraints between those compact representations. We have to note that TDBNs are not restricted to motions, and they can be used with any temporal patterns.

The results from the experiments show that TDBNs not only regenerates learned motions, but also possesses several attractive dynamical properties. In the walking motion generation experiments, we empirically demonstrated the stability of the limit cycle in the hidden state space. The same stability is also seen in the experiment where two legs gradually synchronized. In other experiments, the TDBN smoothly transferred between the limit cycles of walking and running, generating a smooth gait transition motion. Such transition occurs as the result of bifurcation in the system state, triggered by the change of the control parameter, making the transition more robust and flexible. These dynamical behaviors are vital to motion generation in both humans and robots. It is novel to produce synchronization of limbs and hysteresis of a gait transition in motions generated by a neural network trained by real motions.

Acknowledgments

This research is partially supported by the Aihara Project, the FIRST program from JSPS, initiated by CSTP.

References

JJ Collins and IN Stewart. Symmetry-breaking bifurcation: a possible mechanism for 2:1 frequency-locking in animal locomotion. *Journal of mathematical biology*, 30(8):827–838, 1992.

F.J. Diedrich and W.H. Warren Jr. Why change gaits? Dynamics of the walk run transition. *Journal of Experimental Psychology*, 21(1):183–202, 1995.

S. Grillner. Neurobiological bases of rhythmic motor acts in vertebrates. *Science*, 228(4696): 143, 1985.

G.E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002.

G.E. Hinton. Boltzmann machine. http://www.scholarpedia.org/article/Boltzmann_machine, 2007.

G.E. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504, 2006.

G.E. Hinton, S. Osindero, and Y.W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

D.H. Hubel, J. Wensveen, and B. Wick. *Eye, brain, and vision.* Scientific American Library New York, 1988.

A.J. Ijspeert. Central pattern generators for locomotion control in animals and robots: a review. *Neural Networks*, 21(4):642–653, 2008.

A.J. Ijspeert, A. Crespi, D. Ryczko, and J.M. Cabelguen. From swimming to walking with a salamander robot driven by a spinal cord model. *Science*, 315(5817):1416, 2007.

Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *ICML '09: Proceedings of the 26th Annual International Conference on Machine Learning*, pages 609–616. ACM, 2009.

G. Schöner, WY Jiang, and JAS Kelso. A synergetic theory of quadrupedal gaits and gait transitions. *Journal of theoretical Biology*, 142(3):359–391, 1990.

M.L. Severin F.V. Orlovsky G.N. Shik. Control of walking by means of electrical stimulation of the mid-brain. *Biophysics*, 11:756–765, 1966.

G.W. Taylor, G.E. Hinton, and S.T. Roweis. Modeling human motion using binary latent variables. *Advances in Neural Information Processing Systems*, 19:1345, 2007.