Human Activity Recognition Using Deep Recurrent Neural Networks and Complexity-based Motion Features

Woo Young Kwon[1], **Youngbin Park**[1], **Sang Hyoung Lee**[2] **and Il Hong Suh**[1]
Hanyang University, Korea[1] Korea Institute of Industrial Technology, Korea[2]
{wykwon,pa9301,ihsuh}@hanyang.ac.kr zelog@kitech.re.kr

Abstract

Microsoft Kinect can be used for computationally inexpensive acquisition of skeleton tracking in real time. For human activity recognition, it appears to provide an opportunity for researchers to achieve good performance at low cost. However, two issues still remain. Firstly, the Kinect skeleton tracker often captures unnatural skeleton poses, such as discontinuous and vibrated motions, in the presence of self-occlusion. Secondly, there is still a requirement for anyone wishing to understand human behavior to develop high-level features instead of making direct use of a 3D skeleton pose. To this end, we propose a method that is composed of two parts. The first part is to improve the Kinect skeleton under self-occlusion by using deep recurrent neural networks. The second part is to extract features by evaluating the importance of each subsequence of trajectories using a complexity-based measure.

1 Introduction

For human activity recognition, there are two main approaches depending on the characteristics of features [Yao *et al.*, 2011]. The first is the pose-based approach and the second is the appearance-based approach. Pose-based approaches are based on the features that are derived from articulated 3D joint data, while appearance-based approaches are based on the features that can be extracted from video data without explicit human body modeling. Recently, significant successes of Deep Convolutional Neural Networks (ConvNets) on various computer-vision tasks, such as image recognition [Krizhevsky *et al.*, 2012], segmentation [Noh *et al.*, 2015], detection [Sermanet *et al.*, 2013] have been observed. Therefore, some researchers have attempted to apply ConvNets to appearance-based methods. However, in this paper, we focus on the pose-based approaches and investigate a different way to use deep learning techniques for activity recognition.

Pose-based approaches interpret human behavior as a sequence of skeleton poses. An optical motion-capture system is one of the most popular devices for obtaining 3D joint trajectories, and it has an advantage in the sense that the tracked 3D skeleton poses are reliable with respect to viewpoint change. However, this system is expensive, and significant spatial constraints are involved, such as a large set-up in the surrounding environment. A practical alternative is to use a sequence of images captured from monocular or stereo cameras. However, this requires modeling of the human body and developing an extraction algorithm for human poses under realistic imaging conditions, which is an inherently difficult task.

Recently, the Microsoft Kinect RGBD sensor has been widely used thanks to its cheap price and relatively good resolution, frame rate, accuracy, etc. Kinect not only provides inexpensive 3D depth data, but also enables one to obtain 3D human skeleton poses easily using the built-in skeleton tracker. The use of a Kinect sensor provides an opportunity for pose-based researchers to achieve good performance with low cost and less effort. However, although Kinect trackers can achieve a considerable human body tracking performance, they often capture unnatural skeleton poses, such as discontinuous and vibrated motions, in the presence of self-occlusion. This is a common problem among most vision-based sensing systems, and it can lead subsequently to the poor performance of activity recognition.

Another issue for understanding human behavior is that those who want to classify such behavior are required to develop sophisticated features. Much of the earlier work in this area used the skeletal trajectories themselves to represent and recognize human activity. The joint trajectories are used directly as inputs for classifiers, such as the conditional random field, hidden Markov models (HMMs), support vector machines (SVMs). A direct comparison of 3D skeleton poses in space and time is not suitable for attaining good recognition accuracy. This is because semantically similar behaviors may not necessarily imply a numerically similar joint trajectory. Instead, discriminative features are required that are extracted from a static pose or a short sequence of poses.

Recently, deep neural networks that are trained using an end-to-end learning procedure have gained much attention for their ability to solve various machine-learning problems, including object recognition [Krizhevsky *et al.*, 2012], semantic segmentation [Noh *et al.*, 2015], caption generation [Xu *et al.*, 2015], robotic manipulation [Levine *et al.*, 2015], etc. This is due largely to the inherent properties of deep neural networks. However, to the best of our knowledge, there is

no end-to-end deep learning model to achieve promising performance in pose-based activity recognition. This is because most deep learning models require a considerable amount of labeled training data. With a limited number of such data, training a complex model can result in serious overfitting. Indeed, it is extremely difficult to construct an activity recognition dataset that has numerous class labels.

Therefore, in this paper, we restrict the use of deep learning for improving 3D human poses captured by the Kinect skeleton tracker to those cases for which we can easily obtain a large number of labeled training data. As mentioned earlier, this is not sufficient to produce a considerable recognition accuracy. Human actions can be efficiently represented as a set of a small number of subsequences of those improved skeleton trajectories. In this sense, we developed a complexity-based subsequence of the time-series clustering (STSC) method to extract discriminating features. Specifically, a time-series complexity measure is used to evaluate the importance of each subsequence. The selected important subsequences are then clustered to construct motion features for the effective representation of the activities. For the classification of human activity, multivariate Gaussian hidden Markov models (HMMs) are employed. To generate observations for the HMMs, the motion features described above are computed from each joint and they are concatenated. The combined features are then transformed to an observation in the reduced Euclidean space.

2 Related Work

A joint trajectory that is captured from a particular human activity can be represented as a 4-D XYZT space-time volume, where the 3-dimensional (XYZ) points correspond to the joint positions. There are mainly three popular devices for extracting a skeleton pose: multi-camera motion capture (MoCap) systems, monocular or stereo cameras, and Kinect. As mentioned before, some approaches use the trajectories themselves to represent and recognize the actions directly [Yang *et al.*, 1997; Oliver *et al.*, 2002; Wang and Mori, 2009]. However, the majority of studies extract meaningful patterns from the trajectories. This supports the hypothesis that even though the skeleton pose itself contains high-level information for activity recognition, the extraction of features from the raw joint-trajectory data might provide an opportunity to achieve better performance.

The approaches are classified into two groups by means of the ways to construct meaningful features. One approach for recognizing human motions relies on extracting features from each frame, and it considers an action as a sequence of feature vectors. The other approach employs more descriptive features from a joint trajectory. The histogram of 3D joint locations [Xia *et al.*, 2012] leads on to the histogram of spherical coordinates of the joint positions in a coordinate system that uses the hip joint as the origin. The sequence of the most informative joints [Ofli *et al.*, 2014] gives the top 6 according to the variance of joint angle and angular velocity, and allows the construction of feature vectors with the features of these most informative joints. Eigenjoint [Yang and Tian, 2012] employs the position differences between joints to represent human actions. It computes the position difference of all the pairs of joints within one frame, the joints of two consecutive frames, and the joints of one frame and the initial frame, to capture the spatial and temporal configuration of human poses.

As mentioned earlier, deep learning techniques are mostly applied in appearance-based approaches [Wang *et al.*, 2015; Simonyan and Zisserman, 2014; Karpathy *et al.*, 2014]. To overcome the problem of overfitting with a small number of annotated training data, they transform the task of action recognition into one of image classification. The static appearance by itself is a useful clue, since some actions are strongly associated with particular objects and static poses. However, such an approach does not consider the motion information that is encoded in multiple contiguous frames. To make up for this weakness, most studies propose a particular method to effectively incorporate the motion information. ConvNet operates at each time step of the motion trajectory, effectively performing action recognition; the classification results from the whole video frame are then combined for the final decision. In fact, these approaches produce a relatively competitive recognition accuracy.

3 Improving the Kinect Skeleton Using Deep Recurrent Neural Networks

The first part of the proposed model is to improve the joint position and velocity of the Kinect skeleton using supervised learning, and it is implemented based on our prior work [Park *et al.*, 2016]. The inputs for the supervised learning are sequences of 3D positions or velocities obtained by the Kinect skeleton tracker. The targets are sequences of skeleton poses captured using a commercial optical maker-based motion capture system. In our method, a deep recurrent neural network is employed to solve the regression problem, in which two deep recurrent neural networks are trained separately for refining the positions and velocities of the body joints. In this Section, we will briefly describe a deep recurrent neural network, and we present the details of how to train such a network.

3.1 Deep Recurrent Neural Network

A recurrent neural network (RNN) [Williams and Hinton, 1986] is a neural network that simulates a discrete-time dynamical system; it is a powerful model for sequential data. A conventional RNN is constructed by defining the transition function and the output function as

$$\mathbf{h}_t = \phi_h\left(\mathbf{W}^T\mathbf{h}_{t-1} + \mathbf{U}^T\mathbf{x}_t\right) \quad (1)$$

$$\mathbf{y}_t = \phi_o\left(\mathbf{V}^T\mathbf{h}_t\right), \quad (2)$$

where ϕ_h, ϕ_o, $\mathbf{x}_t$, $\mathbf{y}_t$ and $\mathbf{h}_t$ are respectively a state transition function, an output function, an input, an output, a hidden state, and $\mathbf{W}$, $\mathbf{U}$ and $\mathbf{V}$ are the transition, input and output matrices, in that order. It is usual to use a nonlinear function such as a logistic sigmoid function or a hyperbolic tangent function for ϕ_h.

RNNs are inherently deep in time, since their hidden state is a function of all previous hidden states. However, the potential weakness for RNNs is that RNNs lack hierarchical

processing of the input in space. From this perspective view, deep recurrent neural networks has recently gained significant attention to many researchers. As with feedforward deep neural networks have multiple nonlinear layers between input and output, a recurrent network can be considered as a deep recurrent neural network (DRNNs) if the network has more than one hidden layers. We can now consider two schemes of DRNNs. One has L hidden layer with temporal connection only at the l-th layer and the other has L hidden layer with full temporal connections (called stacked RNN). Based on empirical evaluation on our datasets, we have chosen the former scheme. The values of the output units are computed by linear activation.

Because skeleton tracking is an inherently dynamic process, it seems natural to consider DRNNs as a model for supervised learning. As with most related researcher, we considered the two most popular deep learning techniques, dropout and Rectified Linear Units (ReLU) [Krizhevsky *et al.*, 2012] for our initial training of DRNNs. We used a Rectified Linear Unit (ReLU) as the nonlinear activation function for all units in the hidden layers. Unfortunately, dropout does not work well with RNNs unlike with feedforward deep neural networks. Although we applied dropout carefully to the DRNNs with our datasets according to the method proposed by [Zaremba *et al.*, 2014], we found that it leads to divergence.

3.2 Details of Training Two DRNNs

In the following discussion, we will refer to the DRNNs for improving the joint position and velocity of a skeleton as pDRNN and vDRNN, respectively. pDRNN and vDRNN have five layers, three of which are hidden and the remaining two are the input and output, respectively. The number of units in the input and output layers is 48, because the number of joints to be refined is 16 and each joint is composed of 3 (x, y and z) coordinates. Kinect v2 supports 25 joints, and 16 joints are used in our method. These are as follows: *spinebase*, *spinemid*, *neck*, *shoulderleft*, *elbowleft*, *wristleft*, *shoulderright*, *elbowright*, *wristright*, *hipleft*, *kneeleft*, *ankleleft*, *footleft*, *hipright*, *kneeright*, *ankleright*, *footright*, and *spineshoulder*.

Among the full 25 joints, some joints, such as *thumbleft* and *thumbright* are tracked in a very unstable manner. In addition, some joints are not supported by the motion capture system. As a result, *head*, *handleft*, *handright*, *handtipleft*, *thumbleft*, *handtipright*, *thumbright*, *footleft* and *footright* were excluded in our method.

The temporal length of the training data for pDRNN is 7 frames. In the training phase, the inputs are the joint positions of the Kinect skeleton (denoted by **z**), and the targets are the joints tracked using the motion capture system. The temporal length of the training data for vDRNN is 15 frames. The training data for vDRNN are the velocities of the improved skeleton poses, which are defined by $\mathbf{v}_t = \tilde{\mathbf{z}}_t - \tilde{\mathbf{z}}_{t-1}$. We denote the input for vDRNN as **v**. We denote the input for vDRNN as $\tilde{\mathbf{z}}$. The outputs for pDRNN and vDRNN are denoted by $\tilde{\mathbf{z}}$ and $\tilde{\mathbf{v}}$, respectively. The L-BFGS optimization algorithm is used to train the two networks from random initializations, and the sum-of-squared errors are used for the objective functions.

3.3 Integration based on Kalman Filtering

In the Kalman filter framework, the dynamics and the measurements are modeled by the following discrete-time state-space model:

$$\mathbf{x}_t = \mathbf{F}_t\mathbf{x}_{t-1} + \mathbf{G}_t\mathbf{v}_t + \mathbf{w}_t \quad (3)$$

$$\mathbf{z}_t = \mathbf{H}_t\mathbf{x}_t + \mathbf{u}_t. \quad (4)$$

where **x**, **z**, **v**, **F**, **G** and **H** are the state vector, measurement vector, input control vector, state transition matrix, input transition matrix, and measurement matrix, respectively. It is assumed that **w** is the process noise vector, which has a zero mean but with a covariance matrix $\mathbf{Q} = E\{\mathbf{w}\mathbf{w}^T\}$, and **u** is the measurement noise vector that also has zero mean with a covariance matrix $\mathbf{R} = E\{\mathbf{u}\mathbf{u}^T\}$. In this work, since we consider an uncorrelated covariance matrix, **Q** and **R** become diagonal matrices. In our experiment, **F**, **G** and **H** were set equal to the identity matrix, and hence the prediction model becomes $\mathbf{x}_t = \mathbf{x}_{t-1} + \mathbf{v}_t$. Matrices **Q** and **R** were determined by using the validation dataset.

The state $\mathbf{x}_t$ that we are required to estimate is the true skeleton pose, and its dimension is 48 as mentioned earlier. Our contribution is to replace the measurement vector $\mathbf{z}_t$ with the improved body joints $\tilde{\mathbf{z}}$, and the input control vector $\mathbf{v}_t$ with the enhanced velocities $\tilde{\mathbf{v}}_t$. Therefore, the j-th row and j-th column of **R** and **Q** are determined by computing $(z_i^{j,M} - \tilde{z}_i^j)$ and $(v_i^{j,M} - \tilde{v}_i^j)$, respectively. Here, $z^{j,M}$ is the j-th component of the i-th training data obtained from the motion capture system, and $v_i^{j,M}$ represents the true velocity of the j-th component of the i-th frame. In our methods, $\mathbf{x}_0$ was set to equal $\tilde{\mathbf{z}}_0$. The details are described in [Park *et al.*, 2016].

4 Complexity-based Motion Features

4.1 Complexity-based Subsequence of Time-series Clustering

Subsequence of time-series clustering (STSC) is a well-known pattern discovery technique that uses time-series data. In STSC, the time series data are represented as a set of subsequences, and all the subsequences of the time series are extracted using a sliding window. Next, the extracted subsequences are grouped into clusters using a clustering algorithm such as k-means. The obtained cluster centers can be used as motion features. Unfortunately, it has been demonstrated that a typical STSC algorithms produces meaningless results [Keogh and Lin, 2005]. Averaging of all subsequences extracted by a sliding window generates some form of sine waves irrespective of the original shape of the pattern in the input data.

By evaluating the meaningfulness of subsequences and by selectively using these subsequences, the STSC algorithms can produce more meaningful results. In previous work [Kwon and Suh, 2014], a complexity-based STSC method was proposed. In the research, predictive information [Grassberger, 1986], which is a type of time-series complexity measure, is employed to evaluate the meaningfulness of a subsequence in a joint trajectory. Intuitively, a meaningful subsequence requires a considerable amount of information to

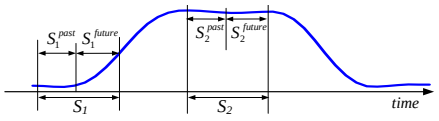

Figure 1: An example of time-series complexity.

describe itself, whereas a meaningless subsequence requires less information. A complexity measure can be used for evaluating the meaningfulness of a subsequence because the complexity can be viewed as a measure of the amount of useful information in a set of data [Rissanen, 2007].

Formally, given a multidimensional time series $\mathbf{X}$ of length T, a subsequence $\mathbf{S}_p$ of $\mathbf{X}$ is a sampling of length $w < T$ of contiguous positions from $\mathbf{X}$, that is, $\mathbf{S}_p = \mathbf{x}_p, \mathbf{x}_{p+w-1}$ for $1 \leq p \leq m-w+1$. Figure 1 illustrates a time series and two of its subsequences $\mathbf{S}_1$ and $\mathbf{S}_2$. Intuitively, $\mathbf{S}_1$ seems to be more complex than $\mathbf{S}_2$ because $\mathbf{S}_1$ requires more information to encode it than does $\mathbf{S}_2$. This difference can be measured using mutual information based on information theory.

Given a subsequence of a multidimensional time series $\mathbf{S} = \mathbf{s}_i, \ldots, \mathbf{s}_{i+m}$, the complexity is modeled as the mutual information between the past and future, and is given by

$$C(\mathbf{S}_i) \equiv I(\mathbf{S}_i^{past}; \mathbf{S}_i^{future}), \tag{5}$$

where $\mathbf{S}_i^{past} = \mathbf{s}_i, \ldots, \mathbf{s}_{i+m/2-1}$, $\mathbf{S}_i^{future} = \mathbf{s}_{i+m/2}, \ldots, \mathbf{s}_{i+m-1}$, and $I(\mathbf{S}_i^{past}; \mathbf{S}_i^{future})$ is the mutual information between $\mathbf{S}_i^{past}$ and $\mathbf{S}_i^{future}$.

Based on the complexity measure, only important subsequences can be used for clustering. Algorithm 1 shows the proposed complexity-based STSC method. Here, the length of the sliding window, w, and complexity threshold, τ are dependent on the dataset. Before clustering, each important subsequence is normalized to have a zero mean by

$$s_i' = s_i(1/w) \sum_1^{i+w-1} s_i. \tag{6}$$

4.2 Subspace mapping of cluster centers

In earlier work [Kwon and Suh, 2014], it was shown that a complexity-based STSC is a useful way to discover clusters of meaningful subsequences. However, this method has the disadvantage that the recognition performance is sensitive to the cluster size. To overcome this problem, we have employed subspace mapping, whereby cluster centers are mapped into a Euclidean subspace preserving pairwise distances.

After the clustering of subsequences, we can obtain a set of cluster centers, $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_K\}$, where K is the number of clusters and $\mathbf{x}_i$ contains the data points of the ith cluster center. The distance between two cluster centers i and j is defined as

$$d_{ij}^2 = (\mathbf{x}_i - \mathbf{x}_j)^T (\mathbf{x}_i - \mathbf{x}_j) \tag{7}$$

We want to find transformed data points $\mathbf{r}_i$ for all i in a reduced space preserving the pairwise distances of cluster centers. This problem is given as an optimization problem by

$$Y^* = \arg\min_Y \sum_{i=1}^{k} \sum_{j=1}^{k} \left[\mathbf{x}_i^T \mathbf{x}_j - \mathbf{y}_i^T \mathbf{y}_j\right]. \tag{8}$$

Algorithm 1 Complexity-based STSC ($\mathcal{D}, \tau, w$)

- $\mathcal{D}$: a dataset of N time series, $\mathcal{D} = \{X_1, \ldots, X_N\}$.
- τ: the threshold value for important subsequences.
- w: the length of subsequences.
- $\mathcal{X}$: a set of high complexity-valued subsequences.

$\mathcal{X} = \emptyset$
for $j = 1$ to N **do** ▷ every time series in $\mathcal{D}$
  $T = t_1, \ldots, t_m \leftarrow \mathcal{D}_j$
  for $i = 1$ to $m - w + 1$ **do** ▷ all subsequences in T
    $S \leftarrow t_i, \ldots, t_{i+m}$
    $X \leftarrow zero\_mean\_normalize(S)$ ▷ as in (6) subsequences in T
    $score = I(X^{Past}; X^{Future})$
    if $score > \tau$ **then** ▷ store only important subsequences
      $\mathcal{X} \leftarrow \mathcal{X} \cup X$
    end if
  end for
end for
$cluster\_centers \leftarrow$ *k-means_clustering*($\mathcal{X}$)
return $cluster\_centers$

As shown in [Kruskal and Wish, 1978], the transformed data points Y^* can be obtained by eigenvalue decomposition of the Gram matrix of X, where $G = X^T X$. By using eigenvalue decomposition, the Gram matrix is given by $G = X^T X = Q \Lambda Q^T$. By using the top n eigenvectors of the Gram matrix and their respective eigenvalues, the matrix of transformed data points, Y^* is given by $Y^* = Q\Lambda^{1/2}$. The gram matrix is computed by using pairwise distances as

$$g_{ij} = (\mathbf{x}_i - \mathbf{x}_1)(\mathbf{x}_j - \mathbf{x}_1) = (d_{1j}^2 + d_{i1}^2 - d_{ij}^2)/2 \tag{9}$$

As a result, the coordinates of cluster centers in a reduced subspace can be obtained. Figure 2 shows the overall processes of complexity-based STSC and subspace mapping.

4.3 Action classification

For the classification of human activity, we employed multidimensional Hidden Markov Models (HMMs) with continuous observation probability, where the probability density function is given by a multivariate Gaussian distribution. A set of parameters of the HMM for the ith class and the jth joint is defined as $\lambda_i^j = \{\Pi_i^j, A_i^j, \mu_i^j, \Sigma_i^j\}$, where Π, A , μ, and Σ are the initial state probabilities, the state transition probabilities, the mean vector of Gaussian observation distribution, and the covariance matrix, respectively.

An observation sequence for each HMM is given by using the complexity-based STSC method and subspace-mapping of cluster centers.

Let $\tau_t^j = \{x_t^j, y_t^j, z_t^j\}$ be jth 3D joint position at time t. The skeletal trajectory of a human motion is given as $\tau_1^j, \tau_2^j, \ldots, \tau_T^j$. The procedure to generate the observation sequences is as follows:

1. Obtain a subsequence by sliding windows as $\mathbf{s}_1^j, \mathbf{s}_2^j, \ldots, \mathbf{s}_{T-w+1}^j$, where w is the length of a sliding window, and $\mathbf{s}_i^j = \tau_i^j, \tau_{i+1}^j, ..., \tau_{i+w-1}^j$

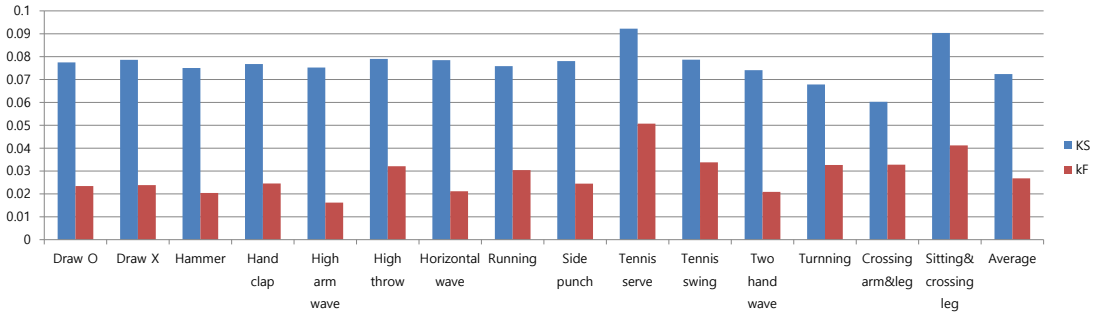

Figure 3: Average position errors produced by the Kalman filter-based proposed method, denoted by kF, and Kinect skeleton, denoted by KS.

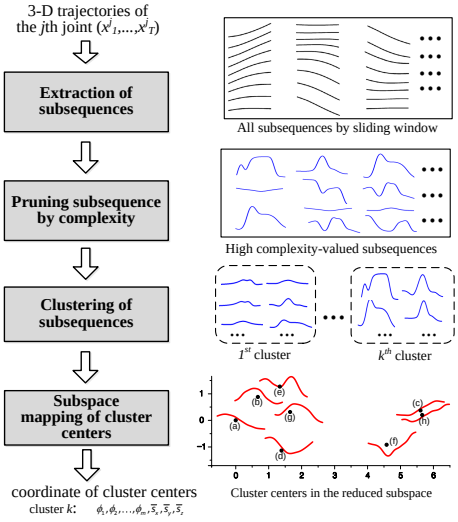

Figure 2: Procedure of the complexity-based STSC features generation.

2. Normalize the subsequence, $\mathbf{s}_i^j$, using the zero-mean normalization in (6).
3. Find the nearest complexity-based codewords k with a given subsequence $\mathbf{s}_i^j$.
4. Obtain the reduced coordinates of the k-th cluster center $\rightarrow \phi_1, \phi_2, ..., \phi_m$
5. Observation: $\mathbf{o}_t^j = \{\phi_1, \phi_2, ..., \phi_m, \bar{s}_x, \bar{s}_y, \bar{s}_z\}$, where $\bar{s}_x$ is the mean value of subsequence s_x.

The probability of an observation sequence $\mathbf{o}$ given a class i is given as

$$P\left(\mathbf{o}|\text{class} = i\right) = P\left(\mathbf{o}|\lambda_i^1\right) P\left(\mathbf{o}|\lambda_i^2\right) \ldots P\left(\mathbf{o}|\lambda_i^J\right). \quad (10)$$

5 Experiments

5.1 Experimental Setup

We used Microsoft Kinect v2 in our experimental configuration. The proposed algorithm has not been evaluated on public datasets, such as MSRAction3D, MSRDailyActivity3D. Instead, we created a dataset composed of 16 activity classes. The activities are *draw O*, *draw X*, *forward punch*, *hammer*, *hand clap*, *high arm wave*, *high throw*, *horizontal wave*, *running*, *side punch*, *tennis serve*, *tennis swing*, *two hand wave*, *crossing arm & leg*, *turning* and *sitting & crossing leg*. Some activities, such as *crossing arms & legs*, *sitting & crossing leg*, consist of a large amount of severe self-occlusion poses, while *two hand wave* and *side punch*, for instance, include a small number of self-occlusion poses. Each activity class was repeated 20 times. Every activity starts with a standing pose, and an activity is composed of approximately 100∼200 frames. Most activities in our dataset are similar to activities in the MSRAction3D dataset. Here, it is noted that we generate our dataset due to the following concern. To enable a deep recurrent neural network to improve the whole-skeleton trajectories performed by various people in the public datasets, a great number of training data captured by a number of different people is required, because people in the public datasets have different body shapes. Indeed, it is quite a difficult task to be carried out by a small-sized laboratory.

Every activity except *turning* was performed while facing the Kinect sensors. For the cases of *turning* , the minimum and maximum orientations relative to the Kinect sensor were -90° and 90°, respectively. We did not allow the Kinect sensor to look at the performer's back because the Kinect skeleton tracker cannot distinguish between front and back. The average distance from the Kinect sensor to the human was approximately 3m, and the height of the Kinect above the ground plane was 130cm.

All skeleton trajectories in the dataset were refined using the proposed method described in Section 3, The original skeleton trajectories were also given for the classification task for the purpose of comparison.

Setup for Improving Kinect Skeleton

For supervised learning and evaluation, we employed an OptiTrack motion capture system to provide a set of ground-truth trajectories. The Kinect sensor and the motion capture system tracked skeleton poses simultaneously with recoding capturing time, hence we can construct sets of input and target data pairs. The Kinect sensor and the motion capture system were extrinsically calibrated using a least-squares solution. In addition, we collected training and validation datasets for improving the Kinect skeleton. The training and validation datasets are composed of free movements that a human can

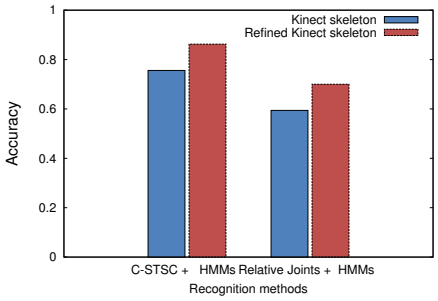

Figure 4: Recognition results on the dataset.

perform, in which behaviors that are similar to the 16 activity classes are included as well. The validation dataset was employed to decide upon the structure of the DRNNs, such as the number of layers, the number of hidden neurons and the temporal length of the training data. The variances of the covariance matrices **R** and **Q** used in the Kalman filtering were determined using the validation dataset as well. The numbers of frames in the training and validation datasets are 17,820 and 5,799 respectively. To construct a dataset to train the deep recurrent neural networks, we sampled sets of data sequences with a temporal stride of 1.

Setup for Activity Recognition

To determine a suitable number of clusters of the complexity-based STSC method, we use the Bayesian Information Criterion (BIC) statistics as described in [Pelleg *et al.*,]. BIC is also used for the number of states in HMMs. The size of the sliding window w and the complexity threshold τ are determined by cross-validation over the training dataset. In the experiment, we use the parameter settings of $w = 36$ and $\tau = -1.0$. For each class of activity, ten actions are used for training, and the rest are used for testing.

5.2 Experimental Results

Average Position Error of Skeletal Trajectory

We compared the Kalman filter-based proposed method, denoted by kF, and the Kinect skeleton, referred to as KS. Figure 3 shows the average position error (APE) of each activity class, and the APE of all the test data. It is observed that kF achieves significantly lower APEs than KS over all activity classes. The APE of all the test data produced by KS is 0.072, whereas kF achieves considerable improvement by decreasing the APE to 0.026.

Activity Recognition

We tested our method on the 16 types of activity classes. As a baseline method for action recognition, a multivariate HMMs model using relative joint features [Lv and Nevatia, 2006] is used. A comparison of the recognition accuracy of the proposed method with the accuracies of the baseline methods is shown in Figure 4. Without refinement of the skeletal joint trajectory, the accuracy of the proposed C-STSC feature achieved 70.0%, while the accuracy of the relative joint feature achieved 59.4%. From this result, it can be seen that the C-STSC feature improves the recognition performance. After refinement of the skeletal joint trajectory using the proposed

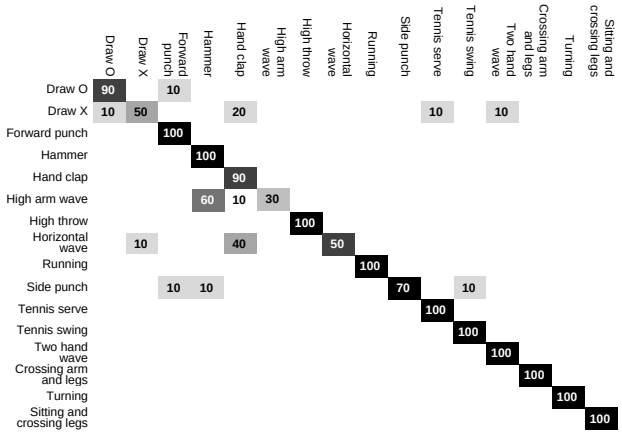

Figure 5: A confusion matrix on the dataset.

method, the accuracies of HMMs with C-STSC features and HMMs with relative joint features are both improved. The result for C-STSC features is 86.5%, and the result for relative joint features is 75.6%. For all the skeletal joint trajectories, the accuracies of the C-STSC feature are higher than the others. The confusion matrix for the result using the C-STSC feature with a refined joint trajectory is illustrated in Figure 5.

6 Conclusions

In this paper, we proposed an improved skeleton tracker and complexity-based motion features for human activity recognition. Our main contributions are to improve the joint position and velocity of the Kinect skeleton, even in the presence of self-occlusion, and then to combine them in a Kalman filter framework. For this, we employed supervised learning with a deep recurrent neural network. Another contribution is the use of a time-series complexity measure for finding important subsequences by using a sliding window on the skeletal trajectories. Furthermore, highly descriptive features can be obtained from the dataset by clustering the important subsequences based on a high measure of complexity.

We showed that the average position error of motion trajectories captured from Kinect is significantly improved by using the proposed method, even when occlusion exists among the skeletal joints. We also showed that the proposed skeleton tracking and the complexity-based motion feature both enhance the accuracy of human activity recognition.

Acknowledgments

This work was supported by the Global Frontier RD Program on <Human-centered Interaction for Coexistence> funded by the National Research Foundation of Korea grant funded by the Korean Government(MEST)(NRFMIAXA003-2010-0029744). This work was also supported by the <Technology Innovation Industrial Program> funded by the Ministry of Trade, (MI, South Korea) [10048320, Technology Innovation Program], by the National Research Foundation of Korea grant funded by the Korea Government (MEST) (NRF-MIAXA003- 2010-0029744)

References

[Grassberger, 1986] Peter Grassberger. Toward a quantitative theory of self-generated complexity. *International Journal of Theoretical Physics*, 25(9):907–938, 1986.

[Karpathy *et al.*, 2014] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.

[Keogh and Lin, 2005] Eamonn Keogh and Jessica Lin. Clustering of time-series subsequences is meaningless: implications for previous and future research. *Knowledge and information systems*, 8(2):154–177, 2005.

[Krizhevsky *et al.*, 2012] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[Kruskal and Wish, 1978] Joseph B Kruskal and Myron Wish. *Multidimensional scaling*, volume 11. Sage, 1978.

[Kwon and Suh, 2014] Woo Young Kwon and Il Hong Suh. Complexity-based motion features and their applications to action recognition by hierarchical spatio-temporal naïve bayes classifier. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),*, pages 3141–3148. IEEE, 2014.

[Levine *et al.*, 2015] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *arXiv preprint arXiv:1504.00702*, 2015.

[Lv and Nevatia, 2006] Fengjun Lv and Ramakant Nevatia. Recognition and segmentation of 3-d human action using hmm and multi-class adaboost. In *Computer Vision–ECCV 2006*, pages 359–372. Springer, 2006.

[Noh *et al.*, 2015] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1520–1528, 2015.

[Ofli *et al.*, 2014] Ferda Ofli, Rizwan Chaudhry, Gregorij Kurillo, René Vidal, and Ruzena Bajcsy. Sequence of the most informative joints (smij): A new representation for human skeletal action recognition. *Journal of Visual Communication and Image Representation*, 25(1):24–38, 2014.

[Oliver *et al.*, 2002] Nuria Oliver, Eric Horvitz, and Ashutosh Garg. Layered representations for human activity recognition. In *Multimodal Interfaces, 2002. Proceedings. Fourth IEEE International Conference on*, pages 3–8. IEEE, 2002.

[Park *et al.*, 2016] Youngbin Park, Sungphill Moon, and Il Hong Suh. Tracking human-like natural motion using deep recurrent neural networks. *arXiv preprint arXiv:1604.04528*, 2016.

[Pelleg *et al.*,] Dan Pelleg, Andrew W Moore, et al. X-means: Extending k-means with efficient estimation of the number of clusters. In *ICML*, pages 727–734.

[Rissanen, 2007] Jorma Rissanen. *Information and complexity in statistical modeling*. Springer Publishing Company, Incorporated, 2007.

[Sermanet *et al.*, 2013] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.

[Simonyan and Zisserman, 2014] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems*, pages 568–576, 2014.

[Wang and Mori, 2009] Yang Wang and Greg Mori. Max-margin hidden conditional random fields for human action recognition. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 872–879. IEEE, 2009.

[Wang *et al.*, 2015] Pichao Wang, Wanqing Li, Zhimin Gao, Jing Zhang, Chang Tang, and Philip Ogunbona. Deep convolutional neural networks for action recognition using depth map sequences. *arXiv preprint arXiv:1501.04686*, 2015.

[Williams and Hinton, 1986] DE Rumelhart GE Hinton RJ Williams and GE Hinton. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[Xia *et al.*, 2012] Lu Xia, Chia-Chih Chen, and JK Aggarwal. View invariant human action recognition using histograms of 3d joints. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*, pages 20–27. IEEE, 2012.

[Xu *et al.*, 2015] Kelvin Xu, Jimmy Ba, Ryan Kiros, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. *arXiv preprint arXiv:1502.03044*, 2015.

[Yang and Tian, 2012] Xiaodong Yang and YingLi Tian. Eigenjoints-based action recognition using naive-bayes-nearest-neighbor. In *Computer vision and pattern recognition workshops (CVPRW), 2012 IEEE computer society conference on*, pages 14–19. IEEE, 2012.

[Yang *et al.*, 1997] Jie Yang, Yangsheng Xu, and Chiou S Chen. Human action learning via hidden markov model. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 27(1):34–44, 1997.

[Yao *et al.*, 2011] Angela Yao, Juergen Gall, Gabriele Fanelli, and Luc J Van Gool. Does human action recognition benefit from pose estimation?. In *BMVC*, volume 3, page 6, 2011.

[Zaremba *et al.*, 2014] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.