CHAPTER 5

Computer Vision for Human–Machine Interaction

Qiuhong Ke*, Jun Liu†, Mohammed Bennamoun*, Senjian An*, Ferdous Sohel‡,*, Farid Boussaid*

*The University of Western Australia, Perth, WA, Australia
†Nanyang Technological University, Singapore
‡Murdoch University, Murdoch, WA, Australia

Contents

Abstract

Human–machine interaction (HMI) refers to the communication and interaction between a human and a machine via a user interface. Nowadays, natural user interfaces such as gestures have gained increasing attention as they allow humans to control machines through natural and intuitive behaviors. In gesture-based HMI, a sensor such as Microsoft Kinect is used to capture the human postures and motions, which are processed to control a machine. The key task of gesture-based HMI is to recognize the

Computer Vision for Assistive Healthcare.
DOI: https://doi.org/10.1016/B978-0-12-813445-0.00005-8

meaningful expressions of human motions using the data provided by Kinect, including RGB (red, green, blue), depth, and skeleton information. In this chapter, we focus on the gesture recognition task for HMI and introduce current deep learning methods that have been used for human motion analysis and RGB-D-based gesture recognition. More specifically, we briefly introduce the convolutional neural networks (CNNs), and then present several deep learning frameworks based on CNNs that have been used for gesture recognition by using RGB, depth and skeleton sequences.

Keywords

Human–machine interaction (HMI), Motion analysis, Gesture recognition, Deep learning

5.1 BACKGROUND OF HUMAN–MACHINE INTERACTION

Human–machine interaction (HMI) refers to the interaction and communication between humans and machines [1]. HMI is a multidisciplinary field, which includes human–robot interaction, human–computer interaction, artificial intelligence, and robotics. HMI is traditionally applied in industrial plants for efficiency, quality, and process safety. Nowadays, HMI is widely used for medical, transportation, and entertainment systems.

5.1.1 Human–Machine Interfaces

Humans interact with machines via interfaces. Traditional user interfaces are physical parts of machines that can be seen and touched. Users have to undergo training before being able to operate the machines. Take the computer as an example. In early digital computers, the interfaces were non-interactive batch interfaces consisting of punch cards. Later on, with the advent of command-line interfaces, users could interactively operate the computers using request–response transactions. In the early 1970s, the first graphical user interface was created to make the interaction between humans and computers more efficient and easy. Compared to the command-line interface, the graphical user interface does not require users to memorize commands [2].

Another type of user interface is invisible natural user interfaces (NUIs). An NUI is a user interface that allows users to rely on their natural and intuitive everyday behavior to perform interactions. Instead of forcing users to learn rules of operation, NUIs focus on understanding the expressions of the users' behaviors [2].

5.1.2 Gesture-Based Human–Machine Interaction

One example of an NUI is gestures. Gesture-based HMI has been used since the 1980s. At that time, humans needed to wear gloves such as the DataGlove and Z-Glove [3] to perform gesture-based interactions. These gloves are used to measure the motion, orientation, and position of the fingers using the sensors inside the gloves.

Gesture-based HMI has been used in military applications. For example, the robot is controlled to mimic the motions of the operator and perform dangerous tasks such as bomb disposal, thus minimizing injuries in these tasks [4]. Gesture-based HMI is also very important in industrial applications. Machines with gesture-based HMI enable users to perform open/close, on/off, and other operations using gestures, which reduces contamination risks and helps cleaning efforts [5]. Gesture-based HMI is now very popular in video game consoles. For example, the Nintendo Wii [6] has a remote controller that includes an infrared camera and accelerometers to track human motion. Bluetooth is used to wirelessly transmit data, thus allowing the users to manipulate objects on the screen using gestures. Similar to the Wii Remote, the Sony PlayStation Move [7] contains a motion controller that tracks the users in three dimensions using an accelerometer, a gyroscope, and a magnetic sensor. The system also contains a video camera to handle the cursor movement, aiming, and other motions. Compared to the Wii and PlayStation Move, the Microsoft Kinect for Xbox 360 [8] is camera-only and allows users to interact with machines without a physical controller.

In this chapter, we focus on the task of gesture-based HMI, which aims to recognize the whole-body human gestures captured by the Kinect. The system of gesture-based HMI consists of the acquisition of human gestures and gesture recognition algorithms. We first briefly explain the data acquisition process using Kinect, and then introduce current deep learning frameworks to solve the gesture recognition task for HMI.

5.2 DATA ACQUISITION FOR GESTURE RECOGNITION

There are two main methods for acquiring data for gesture recognition. One is to use accelerometer-based devices such as data gloves to obtain data for gesture recognition [9–12]. Accelerometer-based devices need complex calibration approaches. Another method is to use vision-based devices to capture human gestures. Vision-based devices are less invasive, more human-centered, and more flexible [13]. The most popular vision-based

device is Kinect due to its ability to acquire different modalities, including RGB, depth, and skeleton sequences [14]. The first version of Kinect uses a structured light method to estimate the depth of the scene. More specifically, the Kinect device contains a near-infrared (NIR) laser projector and an NIR camera. The NIR projector sends a fixed and known dot pattern to the scene. The NIR camera captures the deformed patterns on the surface of the object in the scene, which is then used to estimate the depth information using triangulation techniques. The second generation of Kinect computes the depth information based on the Time-of-Flight (ToF) method, i.e. the depth information is estimated by measuring the time the laser light travels from the projector to the scene. The ToF provides depth information in real time and it is easy to capture human motions for gesture recognition in HMI [15]. Once the depth map is obtained, it can then be used to generate the human skeleton. The skeleton is estimated in two steps [16]: (1) the body part image is estimated from the depth image, and (2) the skeleton is estimated from the body part image. To obtain the body part image from the depth image, a motion capture system is first used to capture 10,000 depth images with known skeletons. The data samples are extended to one million by rendering each image to more images using computer graphics techniques. The samples are used to learn a randomized decision forest that maps the depth images to body parts. The body part images are transformed to a skeleton using the mean shift algorithm. For each frame, 20 skeleton joints are estimated. Each joint is provided with the x, y, and z coordinates.

5.3 COMPUTER VISION-BASED GESTURE RECOGNITION

The RGB, depth, and skeleton sequences obtained by Kinect provide the gesture information of the whole human body. In this section, we introduce RGB-D gesture recognition using computer vision algorithms. The aim of computer vision is to provide computers with the same high-level understanding of videos and images as humans have [17]. Computer vision tasks such as gesture recognition are very challenging due to a number of intra-class variations (e.g. viewpoint differences and temporal variations). Traditional methods focus on designing features such as SIFT [18] and HOG [19] to represent the raw data (i.e. pixels), and then use classifiers to solve the problems. These traditional hand-crafted features rely on expert knowledge and extensive human labor as it is difficult to know what is the best feature. Nowadays, deep learning has achieved great success in computer

vision. The performance of many tasks has been dramatically improved [20]. Compared to traditional methods, deep learning acquires representations directly from the data in a hierarchical way and exploits more useful information. One of the most widely used deep learning networks in the computer vision community is Convolutional Neural Networks (CNNs) [21]. Compared to traditional neural networks with fully connected layers, CNNs are easier to train and are capable of generalizing well [20]. In this section, we introduce deep learning frameworks based on CNNs to learn features of sequences for gesture recognition. Since the methods are based on CNNs, we first briefly introduce some concepts of CNNs, and then present RGB-based, depth-based, and skeleton-based gesture recognition methods using CNNs.

5.3.1 Convolutional Neural Networks

A CNN is made up of multiple layers of neurons, each of which is a nonlinear operation on a linear transformation of the preceding layer's outputs. The layers mainly include convolutional layers and pooling layers. The convolutional layers have weights that need to be trained, while the pooling layers transform the activation using a fixed function.

5.3.1.1 Convolutional Layers

A convolutional layer contains a set of filters whose parameters need to be learned. The height and weight of the filters are smaller than those of the input volume. Each filter is convolved with the input volume to compute an activation map made of neurons. In other words, the filter is slid across the width and height of the input and the dot products between the input and filter are computed at every spatial position. The output volume of the convolutional layer is obtained by stacking the activation maps of all filters along the depth dimension. Since the width and height of each filter is designed to be smaller than the input, each neuron in the activation map is only connected to a small local region of the input volume. In other words, the receptive field size of each neuron is small, and is equal to the filter size. The local connectivity is motivated by the architecture of the animal visual cortex [22] where the receptive fields of the cells are small. The local connectivity of the convolutional layer allows the network to learn filters which maximally respond to a local region of the input, thus exploiting the spatial local correlation of the input (for an input image, a pixel is more correlated to the nearby pixels than to the distant pixels). In addition, as the activation map is obtained by performing convolution

between the filter and the input, the filter parameters are shared for all local positions. The weight sharing reduces the number of parameters for efficiency of expression, efficiency of learning, and good generalization.

5.3.1.2 Pooling Layers

A pooling layer is usually incorporated between two successive convolutional layers. The pooling layer reduces the number of parameters and computation by down-sampling the representation. The pooling function can be max or average. Max pooling is commonly used as it works better [23].

5.3.2 RGB-Based Gesture Recognition

Complex human gestures might last for a long time and contain multiple temporal stages with dynamic human postures and motions. Given a gesture video with multiple frames, the temporal information needs to be exploited for a good understanding of the human gesture. Regular CNNs learn information from a single image and are incapable of learning long-range temporal structures. Their inability to model the long-range information in the videos leads to unexpected results of gesture recognition. Recently, CNNs have been successfully designed to model the temporal information in the videos. In this section, we introduce the two methods for gesture recognition. Both methods consist of two streams of CNNs, i.e. a spatial stream, which is trained using the original RGB frames, and a temporal stream, which is trained using the optical flow images. The optical flow images are computed from the consecutive frames. We first briefly introduce the method of extracting optical flow, and then introduce the temporal segmentation networks [24] and temporal evolution networks [25] for gesture recognition.

5.3.2.1 Extracting Optical Flow

Optical flow represents the motion of the scene context relative to an observer [26]. For a human gesture, a single frame with only a static scene can introduce ambiguity which makes the inference of the gesture class difficult. The motion of human bodies needs to be exploited for a good understanding of the gesture class. There are many methods of estimating the optical flow between two frames, including differential-based, region-based, energy-based, and phase-based methods [27]. The most widely used method is the differential method [28], which is based on the assumption

of image brightness constancy. Given a video sequence, let the intensity of a voxel at position (x, y) of the tth frame be $I(x, y, t)$. According to the brightness constancy assumption, the intensity of the voxel remains the same despite small changes of position and time period. More specifically,

$$I(x, y, t) = I(x + \delta x, y + \delta y, t + \delta t), \tag{5.1}$$

where $(\delta x, \delta y, \delta t)$ is the small change of the movement. $I(x + \delta x, y + \delta y, t + \delta t)$ can be expressed with a Taylor series expansion:

$$I(x + \delta x, y + \delta y, t + \delta t) \approx I(x, y, t) + \frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t. \tag{5.2}$$

Therefore,

$$\frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t \approx 0, \tag{5.3}$$

$$\frac{\partial I}{\partial x}U_x + \frac{\partial I}{\partial y}U_y + \frac{\partial I}{\partial t} \approx 0, \tag{5.4}$$

where $U_x = \frac{\delta x}{\delta t}$, $U_y = \frac{\delta y}{\delta t}$ are two components of the optical flow of the voxel (x, y, t).

Once the optical flow is computed, it can be used to learn video-level representation for gesture recognition. Traditional methods used histograms of optical flow to represent the temporal information [29]. Using the histogram can result in the lose the structure of the data [30]. We represent optical flow as images that are then fed to a CNN to learn high-level temporal features for gesture recognition.

5.3.2.2 Temporal Segmentation Networks

An overall architecture of temporal segmentation networks is shown in Fig. 5.1. The goal of temporal segmentation networks is to model the temporal dynamics of an entire video, and to perform a video-level prediction. The networks are comprised of two streams of CNNs: a spatial stream CNN and a temporal stream CNN. The input of the two streams are snippets consisting of several video frames and optical flow images. The snippets are sampled from multiple temporal segments of an entire video, i.e. a video is divided into several segments and a snippet is randomly sampled from each segment. For each stream, the network outputs a frame-level preliminary prediction for each snippet. The preliminary prediction of all

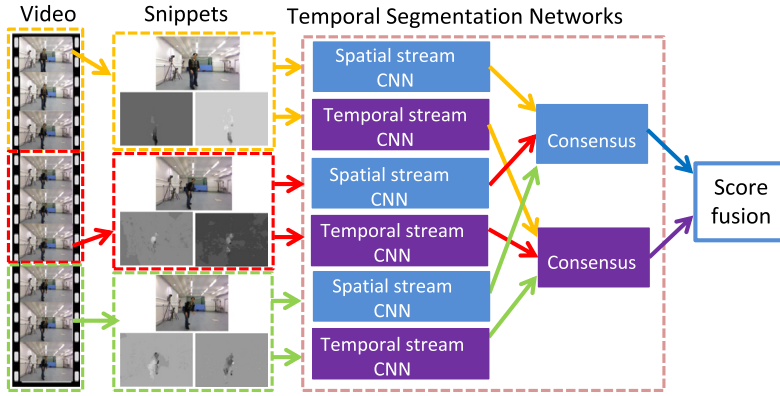

Figure 5.1 Overall architecture of temporal segmentation networks. An entire video is divided into several segments. A snippet is randomly sampled from each segment. The snippets include the original RGB frames and the optical flow images. The frames and the optical flow images are separately fed to the spatial stream and the temporal stream CNNs with weight sharing. The class scores of each snippet are combined using a consensus function to generate a video-level prediction. During testing, the predictions of different modalities are weight averaged for the final determination of the class. (Figure adapted from [24].)

snippets are combined to generate a video-level prediction using a consensus function. During training, the video-level predictions rather than the preliminary frame-level predictions are used to compute the loss value to update the network parameters.

More specifically, given a video D, this method first segments the video into n segments. From each segment, a snippet is randomly selected. Let them be $X_1, X_2, \cdots, X_n$. The snippets are fed to n parallel CNNs with weight sharing. Let the output of the tth CNN be $\mathbf{z}_t = f(W, X_i)$. W represents the parameters of each CNN. The outputs of all segments are preliminary predictions, which are further combined using a consensus function to generate a video-level prediction. In the paper the maximum, weighted averaging, and evenly averaging functions are used as the consensus functions. Of the three, the evenly averaging function achieves the best performance. The video-level prediction is fed to a softmax layer to predict the probability of each class. The network loss is computed as

$$\mathcal{L}(\mathbf{s}, \mathbf{y}) = \sum_{j=1}^{m} y_j \left(\log \sum_{i=1}^{m} \exp s_i - s_j \right), \tag{5.5}$$

where $\mathbf{s} = \frac{1}{n}\sum_{t=1}^{n}\mathbf{z}_t$, which represents the video-level prediction; m is the number of classes; and y_j is the ground truth label for class j.

The network parameters are optimized using a standard back-propagation algorithms. Since the predictions of all snippets are used to jointly update the parameters, the networks learn information from the entire video, thus exploiting the long-term video structure. During training, each video is divided into three segments. Thus, only three frames are used for learning which can reduce the computation cost. The architecture of CNNs are adapted from the Inception with Batch Normalization (BN-Inception) [31] because it has a good balance between efficiency and accuracy [24]. The input of the spatial stream is a single RGB frame image, which is a 3D volume with three channels. For the temporal stream, the input is the combination of optical flow images. More specifically, the optical flow fields x and y computed from every two consecutive frames are transformed into two gray images with values ranging from 0 to 255 using a linear transformation. The two gray images are considered a set of optical flow images. Then 10 sets of optical flow images are stacked as an input. Thus, the input of the temporal stream is a 3D volume of 20 channels.

To avoid overfitting in the training of the deep networks, the ImageNet [32] is used to pre-train the spatial stream CNN. For the temporal stream CNN, cross-modality pre-training is adopted. More specifically, the spatial network is used to initialize the temporal network. Since the input channel of the temporal network is 20, the weights of the first convolutional layer of the spatial network is averaged and repeated for 20 times to handle the input of the temporal stream. In addition to network pre-training, data augmentation is also performed to train the spatial and temporal networks. The augmentation process includes corner cropping, horizontal flipping, and scale jittering [33]. First, each frame image and optical flow image are resized to 256×340. Then five patches are cropped from the four corners and the center with size randomly selected from $[256, 224, 192, 168]$, followed by random horizontal flipping. The cropped patches are then resized to 224×224 for network training.

During testing, 25 RGB frames and 25 stacking optical flow images are first sampled from each video. Each of the RGB frames and optical flow images are then separately fed to the spatial and temporal network to produce frame-level results. For the spatial network and temporal network, the 25 results are averaged before Softmax normalization to output the video-level prediction. Finally, the prediction scores of the spatial stream

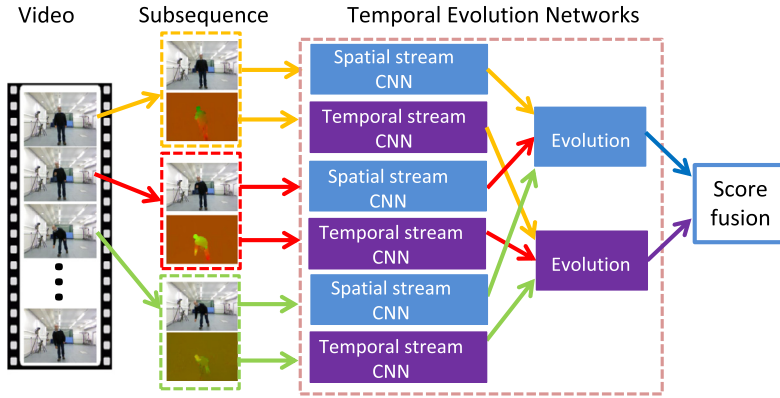

Figure 5.2 Overall architecture of temporal evolution networks. The inputs are sub-sequences of multiple consecutive frames and optical flow images. Each frame and optical flow image is fed to deep networks with weight sharing. The outputs are high-level feature representations, which are then fused in an evolution block to output a sequence-level prediction. During testing, the prediction scores of different modalities are combined to determine the final class. (Figure adapted from [25].)

and temporal stream are averaged as the final prediction to determine the gesture class.

5.3.2.3 Temporal Evolution Networks

Temporal segmentation networks aim to learn the global temporal information of the entire video, while temporal evolution networks [25] focus on leaning the local temporal information. Temporal evolution networks are motivated by the observation that learning the local evolution of human postures provides more useful information than a single frame that contains only static human postures. The inputs of temporal evolution networks are local partial videos. Thus, temporal evolution networks are particularly useful for online recognition and prediction when the entire video is not available.

An overall architecture of the temporal evolution networks is shown in Fig. 5.2. The original temporal evolution is only learned from optical flow images for human interaction prediction [25]. We describe temporal evolution networks which consist of spatial stream and temporal stream CNNs in order to learn both the spatial and temporal information for a better understanding of the video class. The inputs of the two stream CNNs are sub-sequences consisting of multiple consecutive frames and optical flow images. The local sub-sequences contain fine details of human postures and

motions which are very important for gesture recognition. The goal of temporal evolution networks is to use sub-sequences to learn the local evolution of human postures for gesture recognition. Each frame and optical flow image is separately fed to spatial stream and temporal stream networks to learn a frame-level feature representation. The frame-level representations of all the frames are then fused in an evolution block. It includes a temporal convolutional layer to generate a sequence-level representation, followed by a fully connected layer and a softmax layer to output class scores.

More specifically, let an input sub-sequence be $X_1, X_2, \cdots, X_t$. The frame-level feature representation of each frame is $\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_t$, where $\mathbf{u}_i = f(W, X_i)$. W denote the parameters of the CNN. The multiple frame-level feature vectors are then fused with another convolutional layer to output a sequence-level representation $\mathbf{v}$ which is given by

$$\mathbf{v} = \begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_t \end{bmatrix} \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \vdots \\ \gamma_t \end{bmatrix}, \tag{5.6}$$

where $\boldsymbol{\gamma} = [\gamma_1, \gamma_2, \cdots, \gamma_k]^T$ denotes the parameters of the convolutional filter which need to be learned; $\mathbf{v}$ is then fed to a fully connected layer and Softmax layer to generate class scores.

During training, the length of sub-sequence is set to 3 and 7. Among them, 7 achieves the best performance. Longer sub-sequences perform better, but also increase the computational cost. The CNN network is adapted from the CNN-M-2048 network [34]. The input of the spatial stream is the consecutive RGB frame images. For the temporal stream, the optical flow components x and y are combined to a flow coding image by computing the magnitude and the angle of each optical flow vector. Another simple alternative is to set the two components of the optical flow as two channels of an image and to set the third channel to 0. Thus, both the spatial and temporal stream have the same network architecture, which are pre-trained using the ImageNet [32] to avoid overfitting. During testing, the scores of all the sub-sequences are averaged as the video-level prediction. The importance of spatial and temporal information might be different in gesture inference. The weights between the two streams can be learned using ranking SVM to effectively fuse the two models [35].

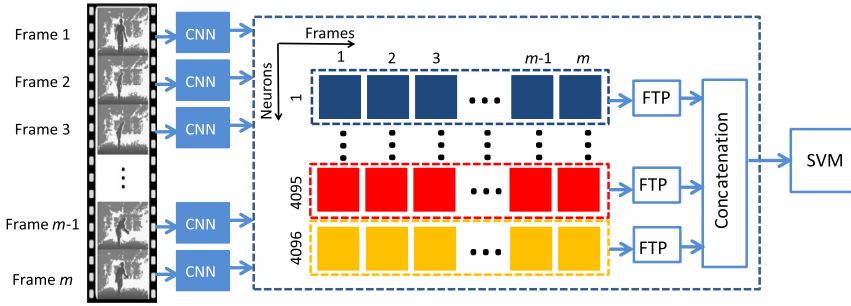

Figure 5.3 Overall architecture of depth-based gesture recognition. Each frame of the depth sequence is fed to a view-invariant CNN which is learned using samples of different viewpoints. The output of the CNN for each frame is a high-level feature vector which is robust to viewpoints. The feature vector of each frame contains 4096 neurons. The time series of each neuron is used to learn Fourier Temporal Pyramid (FTP) features. The FTP features of all neurons are concatenated as the final representation of the video, which is then used to perform classification using SVM. (Figure adapted from [36].)

5.3.3 Depth-Based Gesture Recognition

Given a depth sequence, the RGB-based sequence learning methods (i.e. temporal segmentation network [24] and temporal evolution networks [25]) introduced in Section 5.3.2 can also be used for gesture recognition. In this section, we describe another method which focuses on multiview gesture recognition [36]. An overall architecture of the method is shown in Fig. 5.3. The main idea is to learn a feature representation of each frame using a view-invariant CNN, and then model the temporal structure of the entire video using a Fourier Temporal Pyramid (FTP) [37].

5.3.3.1 View-Invariant Pose Representation

Human gestures might be captured from different viewpoints, which results in large intra-class variations and makes it challenging for gesture recognition. In this section, we describe a view-invariant CNN, which is capable of extracting a high-level view-invariant representation of human postures. The training of the view-invariant CNN requires that the training data contain a large number of viewpoints. Due to the lack of this dataset, a skeleton dataset is used to generate synthetic training data to train the CNN model. More specifically, the skeleton data is rendered from 180 different viewing directions. Each skeleton is fitted with a smooth surface and is then converted to depth images by normalizing the value into the interval between 0 and 255. The generated depth images are then used to train

a deep CNN network. The architecture is adapted from a network used in [38]. The CNN classifies a posture of different viewpoints as the same class in order to learn a high-level view-invariant representation.

5.3.3.2 Temporal Modeling

As shown in Fig. 5.3, given a depth sequence, the first step is to feed each frame to the trained view-invariant CNN to extract a high-level view-invariant representation. The output of the fully connected layer fc7 is used as the feature representation of the input depth image. The feature vector of each frame contains 4096 neurons. The features of all frames produce 4096 time series of neurons. The next step is to feed the time series of each neuron to FTP to learn the temporal information because the representation of each frame contains only the information of the static human posture. The temporal information of the entire video needs to be exploited for gesture recognition.

More specifically, given the CNN features of a depth video, let it be $[\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_t]$; t is the number of frames of the video; $\mathbf{v}_i = [q_{i,1}, q_{i,1}, \cdots, q_{i,m}]^T$ is the CNN feature of the ith frame, which consists of m neurons. The output of the fc7 layer is a 4096D feature vector, thus $m = 4096$. Combing all the frames, the time series of the jth neuron is given by $[q_{1,j}, q_{2,j}, \cdots, q_{t,j}]$. The sequence of each neuron is used for the Fourier transform [39] in hierarchical levels. Take the sequence of the jth neuron as an example. In the first level, the whole sequence $\mathbf{v}_0 = [q_{1,j}, q_{2,j}, \cdots, q_{t,j}]$ is used to perform a Fourier transform and the first p coefficient is used as the descriptor. Let the descriptor of the first level be $\mathbf{d}_{j,1}$. In the second level, $\mathbf{v}_0$ is divided into two segments $\mathbf{v}_{11} = [q_{1,j}, q_{2,j}, \cdots, q_{s,j}]$ and $\mathbf{v}_{12} = [q_{s+1,j}, q_{s+2,j}, \cdots, q_{t,j}]$; $s = [\frac{t}{2}]$; $\mathbf{v}_{11}$ and $\mathbf{v}_{12}$ are separately used to perform the Fourier transform. For each segment, the first p coefficient is used as the descriptor. The descriptors of the two segments are then concatenated as the representation of the second level. Let it be $\mathbf{d}_{j,2}$. Each of the two segments $\mathbf{v}_{11}$ and $\mathbf{v}_{12}$ are then divided into two sub-segments in the next level. Thus, in the third level, there are four segments. As in the previous levels, the four segments are also used to perform a Fourier transform and the first p Fourier coefficient of each segment is extracted. The representation of the third level $\mathbf{d}_{j,3}$ is the concatenation of the descriptors of the four segments. The segment can be divided into sub-segments to extract features. The final representation of the sequence of the jth neuron $\mathbf{d}_j$ is given by the concatenation of the descriptors of all levels $\mathbf{d}_j = [\mathbf{d}_{j,1}, \mathbf{d}_{j,2}, \cdots, \mathbf{d}_{j,l}]^T$; l is the level number; $\mathbf{d}_{j,l} \in R^{c \times 1}$. $c = p \times 2^{l-1}$. All

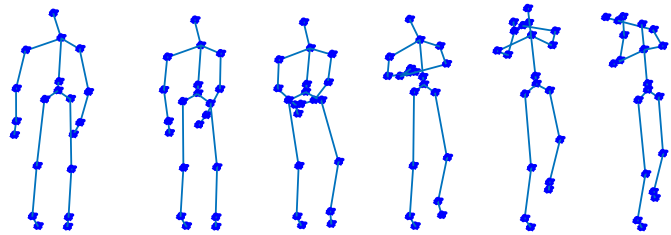

Figure 5.4 Example of a human skeleton sequence (from G3D dataset [45]).

the m neurons are concatenated as the final spatial–temporal representation of the video sequence $\mathbf{d} = [\mathbf{d}_1, \mathbf{d}_j, \cdots, \mathbf{d}_m]^T$. The representations of all the videos can be used for classification using SVM.

5.3.4 Skeleton-Based Gesture Recognition

Compared to the RGB and depth data, the skeleton data only provides the trajectory of discrete human joints. However, skeleton data can also be used for human gesture recognition [40–43]. As shown in Fig. 5.4, the human skeleton joints in each frame depicts the information of five body parts including the trunk, the left hand, the right hand, the left leg, and the right leg. Each part has its own specific structure to form a human posture. In this section, we introduce a SkeletonNet [44] for skeleton-based gesture recognition. The main idea is to extract fine-grained local information of each body part, and then combine the local information of all the parts using CNN to generate a high-level representation for gesture recognition.

5.3.4.1 Robust Features of Body Parts

The structural layout of the five body parts (i.e. the trunk, the left hand, the right hand, the left leg, and the right leg), and the spatial relationship between two body parts is useful information for recognizing a human posture (e.g. the spatial structure of the hand in waving and the relationships of two hands in clapping). Given a human skeleton, any two joints can be connected as a vector (referred to as joint-joint vector). The structural feature of each body part can be described using the geometric relationships between the joint-joint vectors in the same body part, while the spatial relationship between two parts can be represented using the geometric relationships between the joint-joint vectors in the two body parts. As shown in Fig. 5.5, the geometric relationships, including the angle and the magnitude ratios between two joint-joint vectors, are different for two postures.

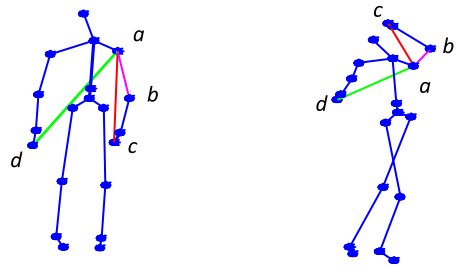

Figure 5.5 Two different human postures (standing and bowling from the G3D dataset [45]) for the two with-part vectors $\mathbf{v}_{ab}$ and $\mathbf{v}_{ac}$ and the two between-part vectors $\mathbf{v}_{ab}$ and $\mathbf{v}_{ad}$. The magnitude ratio and angle between the two vectors are different in the two postures.

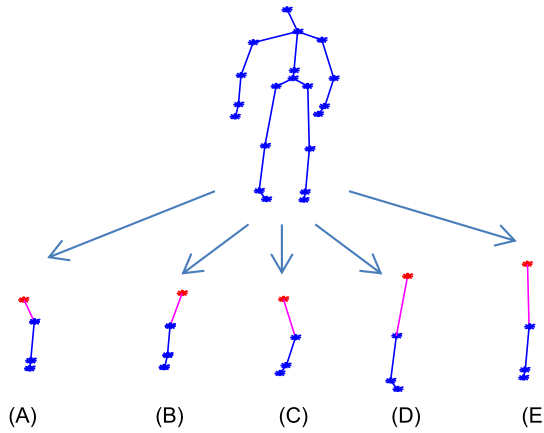

Figure 5.6 A human skeleton is divided into five body parts: (A) the trunk, (B) the left hand, (C) the right hand, (D) the left leg, and (E) the right leg. A starting joint (shown in red) and a reference vector (shown in magenta) are selected for each body part. (Figure adapted from [44].) (For interpretation of the references to color in this figure, the reader is referred to the web version of this chapter.)

The geometric relationships of the joint-joint vectors in the same body part and different body parts depict a human posture, which provides an important clue for gesture recognition. In this section, we introduce robust body features based on the relationships between the joint-joint vectors in the same body part and different body parts.

As shown in Fig. 5.6, a starting joint (shown in red) is selected for each body part, including the head, the left shoulder, the right shoulder, the left hip, and the right hip. The starting joint of each part is connected with

other joints to generate two types of joint-joint vectors: (1) within-part vectors, where the starting joint of each part is connected to the other joints of the same part, and (2) between-part vectors, where the starting joint of each part is connected to the other joints of a different part. A reference vector is also selected in each part, which is shown in magenta in Fig. 5.6. For the kth body part, let the set of within-part vectors be $\mathcal{V}_{\mathrm{w}}^{(k)} \in \mathbb{R}^{m-1}$ and between-part vectors be $\mathcal{V}_{\mathrm{b}}^{(k)} \in \mathbb{R}^{n-1}$; m and n are the numbers of joints in that body part and the whole human body. The cosine distances between all vectors of $\mathcal{V}^{(k)}$ and all vectors of $\mathcal{V}_{\mathrm{w}}^{(k)} \cup \mathcal{V}_{\mathrm{b}}^{(k)}$ are concatenated as the CD feature for the kth body part. The magnitude ratios of all vectors of $\mathcal{V}_{\mathrm{w}}^{(k)} \cup \mathcal{V}_{\mathrm{b}}^{(k)}$ to the reference vector of the kth body part are concatenated as the NM feature for the kth body part.

5.3.4.2 High-Level Feature Learning

Given a skeleton sequence, the CD and NM features of all frames are separately combined as a 2D array with size $p \times t$ for each body part; p denotes the dimension of CD or NM and t is the frame number of the sequence. The CD array and NM array of each body part represents a low-level spatial–temporal information of the sequence. A deep CNN is used to combine the CD and NM arrays of the five body parts. The CNN network is adapted from the CNN-M-2048 network [34], which is pre-trained using the ImageNet [32].

More specifically, the last fully connected layer and output layer of the CNN-M-2048 network [34] are discarded. Each CD and NM array is fed to the CNN model to generate a compact representation. The output feature vectors of the five CD arrays and the five NM arrays are separately concatenated as two feature vectors. The two features are separately fed to two fully connected layers. The two outputs are concatenated to generate a compact representation, which is fed to another fully connected layer and a softmax layer to output the class scores.

5.4 CONCLUSION

In this chapter, we focused on the task of gesture-based HMI and introduced several deep learning frameworks based on CNNs for gesture recognition using the RGB, depth, and skeleton sequences provided by Kinect. One challenge of gesture recognition from sequences is the temporal variance. The long-term temporal structure of the entire sequence is very important to gesture recognition. For RGB sequences, we introduced

temporal segmentation networks [24] and temporal evolution networks [25] to learn the temporal structure of the sequence for gesture recognition. For depth sequences, we introduced a multiview CNN [36] to learn the posture from each frame, followed by a Fourier Temporal Pyramid (FTP) [37] for temporal modeling of the whole sequence for gesture recognition. For skeleton sequences, we introduced robust part features to exploit human postures from each frame and a deep CNN to generate a compact spatial–temporal representation for gesture recognition [44]. The prediction scores of RGB, depth, and skeleton can be fused using simple averaging or a linear SVM for a better performance. The application of gesture-based HMI requires both accuracy and efficiency of gesture recognition. Future works are needed for faster recognition models.

ACKNOWLEDGMENTS

This work was partially supported by Australian Research Council grants DP150100294, DP150104251, and DE120102960.

REFERENCES

[1] G. Johannsen, Human–machine interaction, in: Control Systems, Robotics and Automation, Volume XXI: Elements of Automation, 2009, p. 132.
[2] B. Blazica, The inherent context awareness of natural user interfaces: a case study on multitouch displays, Informatica 38 (4) (2014) 385.
[3] T.G. Zimmerman, J. Lanier, C. Blanchard, S. Bryson, Y. Harvill, A hand gesture interface device, in: ACM SIGCHI Bulletin, vol. 18, ACM, 1987, pp. 189–192.
[4] Robo Sally – bomb disposal robot, https://www.xsens.com/customer-cases/robo-sally-bomb-disposal-robot/ (Accessed 2 March 2017).
[5] https://www.renesas.com/en-sg/about/web-magazine/edge/solution/22-hmi-gesture-recognition.html (Accessed 2 March 2017).
[6] http://gizmodo.com/294642/unearthed-nintendos-pre-wiimote-prototype (Accessed 2 March 2017).
[7] https://en.wikipedia.org/wiki/PlayStation_Move#cite_note-E3_2010_PR-1 (Accessed 2 March 2017).
[8] https://en.wikipedia.org/wiki/Kinect (Accessed 2 March 2017).
[9] E.S. Choi, W.C. Bang, S.J. Cho, J. Yang, D.Y. Kim, S.R. Kim, Beatbox music phone: gesture-based interactive mobile phone using a tri-axis accelerometer, in: IEEE International Conference on Industrial Technology, ICIT 2005, IEEE, 2005, pp. 97–102.
[10] J. Wu, G. Qiao, J. Zhang, Y. Zhang, G. Song, Hand motion-based remote control interface with vibrotactile feedback for home robots, International Journal of Advanced Robotic Systems 10 (6) (2013) 270.
[11] J.K. Min, B. Choe, S.B. Cho, A selective template matching algorithm for short and intuitive gesture UI of accelerometer-builtin mobile phones, in: Second World Congress on Nature and Biologically Inspired Computing (NaBIC), 2010, IEEE, 2010, pp. 660–665.

[12] J. Rao, T. Gao, Z. Gong, Z. Jiang, Low cost hand gesture learning and recognition system based on hidden Markov model, in: Second International Symposium on Information Science and Engineering (ISISE), 2009, IEEE, 2009, pp. 433–438.
[13] Z. Ren, J. Meng, J. Yuan, Depth camera based hand gesture recognition and its applications in human–computer-interaction, in: 8th International Conference on Information, Communications and Signal Processing (ICICS), 2011, IEEE, 2011, pp. 1–5.
[14] I. Ben Abdallah, Y. Bouteraa, C. Rekik, Kinect-based sliding mode control for Lynxmotion robotic arm, Advances in Human-Computer Interaction 2016 (2016) 1.
[15] H. Sarbolandi, D. Lefloch, A. Kolb, Kinect range sensing: structured-light versus time-of-flight Kinect, Computer Vision and Image Understanding 139 (2015) 1–20.
[16] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, R. Moorse, Real-time human pose recognition in parts from single depth images, Communications of the ACM 56 (1) (2013) 116–124.
[17] T. Huang, Computer Vision: Evolution and Promise, Report, European Organization for Nuclear Research, CERN, 1996, pp. 21–26.
[18] D.G. Lowe, Distinctive image features from scale-invariant keypoints, International Journal of Computer Vision 60 (2) (2004) 91–110.
[19] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, CVPR 2005, IEEE, 2005, pp. 886–893.
[20] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444.
[21] D. Graupe, Deep Learning Neural Networks: Design and Case Studies, World Scientific Publishing Co. Inc., 2016.
[22] D.H. Hubel, T.N. Wiesel, Receptive fields and functional architecture of monkey striate cortex, Journal of Physiology 195 (1) (1968) 215–243.
[23] https://en.wikipedia.org/wiki/Convolutional_neural_network (Accessed 2 March 2017).
[24] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, L. Van Gool, Temporal segment networks: towards good practices for deep action recognition, in: European Conference on Computer Vision, Springer, 2016, pp. 20–36.
[25] Q. Ke, M. Bennamoun, S. An, F. Boussaid, F. Sohel, Human interaction prediction using deep temporal features, in: European Conference on Computer Vision Workshops, Springer, 2016, pp. 403–414.
[26] S. Akpinar, F.N. Alpaslan, Video action recognition using an optical flow based representation, in: Proceedings of the International Conference on Image Processing, Computer Vision, and Pattern Recognition (IPCV), The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2014, p. 1.
[27] J.L. Barron, D.J. Fleet, S.S. Beauchemin, Performance of optical flow techniques, International Journal of Computer Vision 12 (1) (1994) 43–77.
[28] A. Bruhn, J. Weickert, C. Schnörr, Lucas/Kanade meets Horn/Schunck: combining local and global optic flow methods, International Journal of Computer Vision 61 (3) (2005) 211–231.
[29] R. Chaudhry, A. Ravichandran, G. Hager, R. Vidal, Histograms of oriented optical flow and Binet–Cauchy kernels on nonlinear dynamical systems for the recognition of human actions, in: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2009, IEEE, 2009, pp. 1932–1939.

[30] Q. Ke, Y. Li, Is rotation a nuisance in shape recognition? in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 4146–4153.
[31] S. Ioffe, C. Szegedy, Batch normalization: accelerating deep network training by reducing internal covariate shift, arXiv preprint, arXiv:1502.03167, 2015.
[32] J. Deng, W. Dong, R. Socher, L.J. Li, K. Li, L. Fei-Fei, ImageNet: a large-scale hierarchical image database, in: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2009, IEEE, 2009, pp. 248–255.
[33] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint, arXiv:1409.1556, 2014.
[34] K. Chatfield, K. Simonyan, A. Vedaldi, A. Zisserman, Return of the devil in the details: delving deep into convolutional nets, in: British Machine Vision Conference, 2014, arXiv:1405.3531.
[35] Q. Ke, M. Bennamoun, S. An, F. Bossaid, F. Sohel, Leveraging structural context models and ranking score fusion for human interaction prediction, arXiv preprint, arXiv:1608.05267, 2016.
[36] H. Rahmani, A. Mian, 3D action recognition from novel viewpoints, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 1506–1515.
[37] J. Wang, Z. Liu, Y. Wu, Learning actionlet ensemble for 3D human action recognition, in: Human Action Recognition with Depth Cameras, Springer, 2014, pp. 11–40.
[38] S. Gupta, R. Girshick, P. Arbeláez, J. Malik, Learning rich features from RGB-D images for object detection and segmentation, in: European Conference on Computer Vision, Springer, 2014, pp. 345–360.
[39] V.O. Alan, W.S. Ronald, R. John, Discrete-Time Signal Processing, Prentice Hall Inc., New Jersey, 1989.
[40] J. Liu, G. Wang, P. Hu, L.Y. Duan, A.C. Kot, Global context-aware attention LSTM networks for 3D action recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017.
[41] J. Liu, A. Shahroudy, D. Xu, G. Wang, Spatio-temporal LSTM with trust gates for 3D human action recognition, in: European Conference on Computer Vision, Springer, 2016, pp. 816–833.
[42] A. Shahroudy, J. Liu, T.T. Ng, G. Wang, NTU RGB+D: a large scale dataset for 3D human activity analysis, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 1010–1019.
[43] Q. Ke, M. Bennamoun, S. An, F. Sohel, F. Boussaid, A new representation of skeleton sequences for 3D action recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017.
[44] Q. Ke, M. Bennamoun, S. An, F. Sohel, F. Boussaid, SkeletonNet: mining deep part features for 3D action recognition, IEEE Signal Processing Letters (2017).
[45] V. Bloom, V. Argyriou, D. Makris, Hierarchical transfer learning for online recognition of compound actions, Computer Vision and Image Understanding 144 (2016) 62–72.