Chapter 16
Non-intrusive Gesture Recognition in Real *Companion* Environments

Sebastian Handrich, Omer Rashid, and Ayoub Al-Hamadi

Abstract Automatic gesture recognition pushes Human-Computer Interaction (HCI) closer to human-human interaction. Although gesture recognition technologies have been successfully applied to real-world applications, there are still several problems that need to be addressed for wider application of HCI systems: Firstly, gesture-recognition systems require a robust tracking of relevant body parts, which is challenging, since the human body is capable of an enormous range of poses. Therefore, a pose estimation approach that identifies body parts based on geodetic distances is proposed. Further, the generation of synthetic data, which is essential for training and evaluation purposes, is presented. A second problem is that gestures are spatio-temporal patterns that can vary in shape, trajectory or duration, even for the same person. Static patterns are recognized using geometrical and statistical features which are invariant to translation, rotation and scaling. Moreover, stochastical models like Hidden Markov Models and Conditional Random Fields applied to quantized trajectories are employed to classify dynamic patterns. Lastly, a non-gesture model-based spotting approach is proposed that separates meaningful gestures from random hand movements (spotting).

16.1 Introduction

Multimodal human behavior analysis in modern human computer interaction (HCI) systems is becoming increasingly important, and it is supposed to outperform the single modality analysis. Like other modalities, for instance facial expression and prosody, gestures play an important role, since they are very intuitive and close to natural human-human interaction. Gestures are spatio-temporal patterns which can be categorized into dynamic patterns (hand movements), static morphs of the hands (postures), or a combination of both. Although gesture recognition technologies have been successfully applied to real-world applications, there are still three major problems that need to be addressed for wider applications of HCI systems: Firstly,

S. Handrich (✉) • O. Rashid • A. Al-Hamadi
Otto-von-Guericke University Magdeburg, Magdeburg, Germany
e-mail: sebastian.handrich@ovgu.de; omer.ahmad@ovgu.de; ayoub.al-hamadi@ovgu.de

S. Biundo, A. Wendemuth (eds.), *Companion Technology*, Cognitive Technologies,
DOI 10.1007/978-3-319-43665-4_16

a gesture-recognition system requires a robust detection and tracking of the relevant body parts, in particular the hands and arms. Since the human body is capable of an enormous range of poses and due to further complexities like self-occlusions and appearance changes, this remains a challenging task. The second major problem is caused by the fact that gesture may vary in shape, trajectory or duration for different persons, or even for the same person when a gesture is performed multiple times. Therefore, features and classifiers must be employed that are invariant to these changes. Furthermore, in a real-world scenario, there are often a lot of random hand movements. The number of these non-gestures is practically infinite and therefore they cannot all be learned in order to be separated from meaningful gestures. In this chapter, the proposed approach is fragmented into two parts. In the first part, the human body parts are detected, which utilizes torso fitting, geodesic distance and path assignment algorithms and builds the basis for the human pose estimation. From the extracted human body parts, the detected hands are utilized in the second part to extract meaningful information for gesture recognition. Keeping in view these goals, Sect. 16.2 presents the literature review of human pose estimation and hand gesture recognition. Section 16.3 is dedicated for the extraction of human poses whereas hand gesture recognition is presented in Sect. 16.4. The experimental results are presented in Sect. 16.5 and conclusion is presented in Sect. 16.6.

16.2 Related Work

16.2.1 Human Pose Estimation

The estimation of the human pose and the robust detection of gesture relevant body parts like hands and arms builds the basis for most gesture recogntion systems. Over the last decade, much research has been done in the area of hand-detection and tracking or even recovering the complete human pose from color or depth cameras. There are several criteria by which these methods can be classified. One of these criteria is whether 3D-data is available or not. Pure image-based methods are usually based on features like skin color [5, 20], contours[14], or silhouettes [4], but often lack the ability to resolve ambiguities, in particular when body parts occlude each other. Sometimes, this problem was addressed by the use of markers [27] (e.g. gloves). Since the aim of the gesture recognition is to push the HMI-systems closer to natural human-human-interactions, the wearing of markers appears to be too cumbersome, but still has its use, for example in the generation of ground-truth data. With the emergence of 3D-sensors like the Microsoft Kinect which are capable of providing dense depth maps in real time, these ambiguities did not disappear but could be more easily resolved.

Human pose estimation methods can be further divided into generative, discriminative and hybrid ones. Discriminative approaches [10, 17, 24] try to match several extracted features to a set of previously learned poses. For this, machine-learning methods like neural networks, support-vector-machines and decision-trees are used. The authors of [24], for example, have used decision forests to determine for each pixel of a depth image the respective body part using a Kinect sensor. This requires a very large training data set (up to 500K images) and the training of the classifiers is computationally very expensive, but attempts to reduce the complexity have been made [3]. Multiple authors have examined the accuracy and robustness of the Kinect's pose estimation and skeleton joint localization. An overview can be found in [11]. In realistic experimental setups, containing general poses, the typical error of the Kinect skeletal tracking was about 10 cm [16]. Another problem of discriminative models is that the pose estimation is often limited to poses that have previously been learned, e.g. [25], where the authors match synthesized depth images of a pose database to observe depth images, where detected hand and head candidates are employed to reduce the size of the search space.

In contrast, generative approaches [18, 21, 23] try to fit an existing model of the human body, in particular a skeleton, to the observed data. In general these approaches are capable of detecting arbitrary poses. However, an initial calibration for the body model is often required (usually a T-pose) and problems can occur when motions are abrupt and fast. For example, the authors of [23] proposed an approach that combines geodesic distances and optical flow in order to segment and track several body parts. The final pose is obtained by adapting a skeleton model consisting of 16 joints to the body part locations. Reported joint errors were in the range of 7–20 cm. A hybrid approach was proposed in [9], where the human pose is estimated from sequences of depth images. In the generative part, a local model-based search is performed that tries to update the configuration of a skeleton model. If this fails, discriminatively trained patch classifiers are used to localize body parts and re-initialize the skeleton configuration. The authors reported average pose errors between 5 and 15 cm.

16.2.2 Gesture Recognition

Gesture recognition is one of the important research domains of computer vision, where the detection, feature modeling and classification of patterns (i.e. gestures drawn by hand) are the major building blocks of the recognition system. Briefly speaking, in gesture recognition, meaningful patterns are identified in the continuous video streams defining the gestural actions. Practically, this is a difficult problem due to the interferences caused by hand-hand or hand-face occlusion, lighting changes or background noise, which cause distortion in the detected patterns and lead to the misclassification of gestural actions. The researchers have utilized various types of image features such as hand color, silhouette, motion and their derivatives. However, the literature presented here is confined to the appearance-based approaches as follows:

Yeo et al. [30] proposed an approach for the hand tracking and gesture recognition in a gaming application. In their approach, after the skin segmentation, the features (i.e., position, direction) are measured for the hand gesture recognition, where the fingertips are utilized to compute the orientation in consecutive frames. Afterwards, a Kalman filter is employed to estimate the optimal hand positions, which results in smoother hand trajectories. The experimental results are reported for the hand gesture dataset, which includes open and closed palm, claw, pinching and pointing. Similarly, Elmezain et al. [7] presented an approach for the isolated and meaningful gestures with the hand location as its feature using HMM. In their approach, the isolated gestures are tested on the number from 0 to 9, which are then combined to recognize the meaningful gesture based on zero-codeword with constant velocity motion. Finally, the experimental results are reported with the average recognition rate of 98.6% and 94.29% respectively for isolated and meaningful gestures.

Wen et al. [28] proposed an approach to detect the hand and recognize the gestures by utilizing image and depth information. In their approach, first the skin color is employed along with K-means clustering to segment the hands and this is followed by extraction of contour, convex hull and fingertip features to represent the gestures. Similarly, Nanda et al. [15] proposed an approach for hand tracking in cluttered environments using 3D depth data captured by Time of Flight (TOF) sensors. The hand and face are detected by applying the distance transform and k-component-based potential fields by calculating weights. The experiments are conducted on a dataset that comprises ten people in which the hands are tracked to recognize the gestures under occlusion. However, the suitability of the proposed approach is affected due to hand shape variation in real scenarios.

Using TOF cameras paired with an RGB camera, Van den Bergh et al. [26] proposed a method for gesture recognition. In their approach, the background is eliminated and the hand is detected by applying a skin segmentation approach. Further, the average 15 Neighborhood Margin Maximization transformation is measured to build the classifier for gesture recognition, where the Haarlet coefficients are calculated to match hand gestures stored in a database. The experiments are presented for both cameras, where the accuracy of 99.2% is acquired with depth-based (i.e., from the TOF cameras) hand detection whereas the accuracy of color-based detection is 92%. The accuracy of gesture recognition is 99.54% and 99.07% for RGB and depth images respectively. Similarly, Yang et al. [29] recognized gestures for the application of a media player. In their approach, eight gestures are recognized based on the hand trajectory features through HMM. Moreover, the tracking of hand is carried out using a continuously adaptive Mean-shift algorithm by taking the depth probability and updates its state using depth histogram. Raheja et al. [19] proposed a Kinect-based recognition system for the hand palm and fingertips tracking in which the depth defines the thresholding criterion for hand segmentation. Finally, the palm and fingertips are determined using the image differencing methods. Beh et al. [2] proposed a gesture recognition system by modeling spatio-temporal data in a unit-hypersphere space approach; a Mixture of von Mises-Fisher (MvMF) Probability Density Function (PDF) is incorporated into HMM. Further, the modeling of trajectories on a unit-hypersphere addresses the constraints of the user's arm length or distance between the user and camera. The results are presented with public datasets, InteractPlay and UCF Kinect datasets, with superior recognition performance compared to relevant state-of-the-art techniques.

To summarize, the problem of gesture recognition is addressed using a wide range of approaches which is impossible to cover here. In fact, the gesture recognition problem is highly context-sensitive, depending on indoor or outdoor environment, types of gesture (i.e. simple to complex gestural commands), single or multiple interaction mediums and different types of sensor. Due to this, it is difficult to generalize the methodologies suggested for different scenarios like gaming applications, air drawn gestures, and pointing gestures. In this chapter, we have investigated in particular the approaches utilizing appearance characteristics and contributes to the feature extraction phase by modeling the features for hand gesture classification.

16.3 Human Pose Estimation

In this work, we propose a generative method for estimating the human pose from depth images. Our approach is based on the measurement of geodesic distances along the surface of the human body and the extraction of geodesic paths to several key feature points. Finally, a hierarchical skeleton is adapted in order to match these paths (Fig. 16.1). The sequences of depth images $\{\boldsymbol{D}_t\}$ are either captured by a depth sensor or synthetically created, depending on the experimental setup. We assume that in each image the user is present and that he is the only object in the scene; so segmentation of the person is beyond the scope of this work. Based on a pinhole camera model, we further compute a 3D point-cloud $\mathbf{W} = \{\mathbf{w}_i \in \mathbb{R}^3\}_{i=1}^{n_x \times n_y}$ from each depth image with resolution $n_x \times n_y$ as follows:

$$\mathbf{w}_i = \begin{pmatrix} -d_i(x_i - c_x)/c_k \\ -d_i(y_i - c_y)/c_k \\ d_i \end{pmatrix}, \tag{16.1}$$

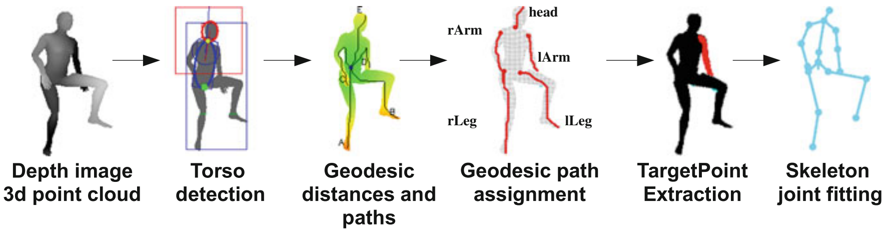


Fig. 16.1 Flow chart of the proposed approach for human pose estimation

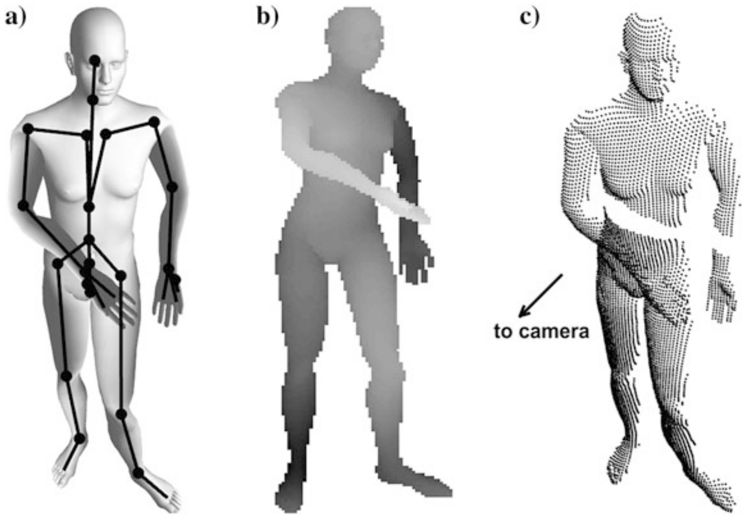


Fig. 16.2 (**a**) Recorded skeleton used to animate a 3D-character. (**b**) Rendered synthetic depth image. (**c**) 3D-point cloud computed from the depth image assuming an ideal pinhole camera model

where $(x_i, y_i)^T$ and d_i are the position and depth of each depth image pixel, respectively, and c_k denotes the camera constant. The image distortion is neglected and the principal point $(c_x, c_y)^T$ is set to the center of the depth image. From here on, the projected 2D position of a 3D point $\mathbf{w}_i$ will be denoted by $\mathbf{w}'_i$.

In order to create synthetic data, we have recorded the skeleton configurations of a person using a Kinect sensor. Using a linear skinning technique, the mesh of a 3D human character model is animated according to the skeleton configuration in each frame and subsequently into a depth buffer (OpenGL). This is shown in Fig. 16.2.

16.3.1 Torso Fitting

Our approach starts by detecting the location of some key-feature points ℓ_j, namely the approximate positions of the torso center, the left and right shoulder, the left and right hip, and the neck. For this, we need at first to segment the torso region. The segmentation is based on the distance transformation, which provides for each pixel of the depth image the 2D-distance to its closest background pixel (Fig. 16.3a). A depth pixel at position $(x, y)^T$ is then considered to be within the torso region $\mathscr{T}$ if its distance value is below a threshold. The threshold value is empirically set to the half of the maximum distance value (Eq. (16.2)). We further compute the centroid of the segmented torso region $\overline{\mathscr{T}}$ and its orientation (Fig. 16.3b). The latter is given by the direction of the largest eigenvector U of the covariance matrix Σ of the torso

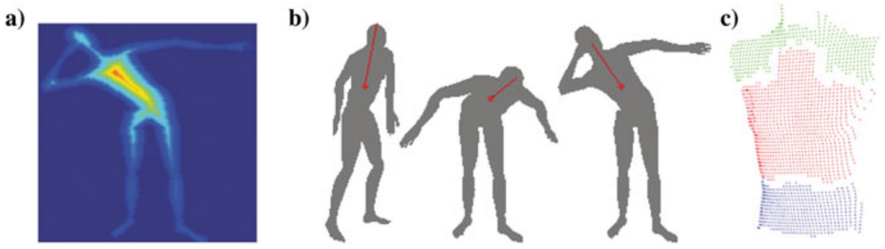


Fig. 16.3 (**a**) Distance transformation for the torso segmentation. (**b**) Initial centroid $\overline{\mathscr{T}}$ (*red dot*) and orientation U (*red line*) of the segmented torso region for several poses. (**c**) 3D torso model

region (Eq. (16.3)).

$$\mathscr{T} = \{(x, y)^T | \; I_{DT}(x, y) < 0.5 \max(I_{DT})\}. \tag{16.2}$$

$$\Sigma = \frac{1}{|\mathscr{T}|} \cdot (\mathscr{T} - \bar{\mathscr{T}}) \cdot (\mathscr{T} - \bar{\mathscr{T}})^T. \tag{16.3}$$

Our next goal is to fit a static 3D-torso model (Fig. 16.3c) to the observed 3D-points within the segmented torso region. Here, static means that only rigid body transformation, i.e. translation, rotation and scaling, are applied. The torso model is initially translated and rotated to match with the centroid $\overline{\mathscr{T}}$ of the torso region and its orientation (given by the eigenvector U). Using the point cloud registration algorithm described in [13], the transformation of the torso model is further refined. This requires the determination of a corresponding point set $C = \{(\mathbf{x}_i, \mathbf{y}_i)^T, \mathbf{x}_i \in \mathbf{X}, \mathbf{y}_i \in \mathbf{Y}\}$ between the torso model 3D points $\mathbf{X}$ and the observed 3D points $\mathbf{Y}$ located in the segmented torso region. For this purpose, we determine for each model point its closest point in the torso region and vice versa. Only points that match each other and are less then 0.1 m apart, are selected as corresponding points. In each frame, the transformation of the torso model in the previous frame is re-used. However, if the residual error $r = \sum_{(\mathbf{x}_i, \mathbf{y}_i)^T \in C} ||\mathbf{x}_i - \mathbf{y}_i||_2$ is too large, the torso fitting is re-initialized using the aforementioned distance transform-based method. Let $\mathbf{T}_{tor}$ denote the determined transformation matrix of the torso model; the positions of the key-feature points are computed by

$$\ell_j = \mathbf{T}_{tor} \mathbf{T}_j \; (0 \; 0 \; 0 \; 1)^T, \tag{16.4}$$

where T_j are transformation matrices specifying the relative transformations of the shoulder, neck, and hip joints in the skeleton model.

16.3.2 Geodesic Distance Measurement

A geodetic distance is the length of a geodesic path, which in turn represents the shortest connection between two nodes of a graph. As described below, such a graph can be constructed from the point-cloud vertices W. Thus, a geodesic path denotes the shortest connection between two arbitrary points along the surface of the human body. The idea of using geodesic distances for pose estimation is motivated by the fact that the geodesic distances of distinct body parts are invariant to pose changes. For example, when the user moves his hand, the Euclidean distance between the hand and the torso center changes, but the geodesic distance does not. However, different body parts may have identical geodetic distances to a certain root point. In order to distinguish between them, the geodesic distances to several points are computed. These points are the key-feature points obtained from the torso fitting step.

A graph consists of nodes and edges $\mathscr{G} = (\mathscr{N}, \mathscr{E})$. The nodes are the 3D-points of the point cloud W. Two criteria are introduced that determine whether two nodes are connected by an edge or not. The first criterion connects two nodes (3D-points) if the Euclidean distance between them is below a threshold. The threshold ϵ_{c1} depends on the resolution of the depth sensor. Typically, we have used 2 cm. Instead of comparing each 3D-point to all others, we make use of the fact that W is an organized point-cloud, i.e. for each 3D-point there is a unique 2D-point in the depth image. We then only compare 3D-points that correspond to adjacent depth pixels. A 3×3 neighborhood is used, so each node can be connected to at most eight others. Using only this criterion, a complete graph can only be generated when there are no self-occlusions (like in a T-pose). Otherwise, the occluding body parts separate individual graph regions from each other. Therefore, a second criterion is introduced that connects two nodes if they are only separated by foreground points, i.e. points that are closer to the camera and as such have a lower depth value. This criterion is only applied to nodes that have not already been fully connected by the first criterion. Furthermore, an edge is only created if neither of the two nodes has already been connected in the direction of the other. For example, a node missing a connection in the left direction (2D) can only be connected to a node that has no connection in the right 2D-direction.

The first edge criterion is given by Eq. (16.5). Here, $\mathbf{w}_i, \mathbf{w}_j \in W$ are two 3D points of the point cloud, $\mathbf{w}'_i, \mathbf{w}'_j$ their corresponding positions in the depth image and $d() = ||\mathbf{w}'_i - \mathbf{w}'_j||_2$ the spatial distance between them:

$$\mathscr{C}_1(i,j) = ||\mathbf{w}_i - \mathbf{w}_j||_2 \le \epsilon_{c1} \wedge d(\mathbf{w}'_i, \mathbf{w}'_j) \le 1. \tag{16.5}$$

The second criterion is defined by Eq. (16.6), where $\overline{\boldsymbol{w}'_i\boldsymbol{w}'_j}$ is the set of all depth pixels located at the straight line between $\boldsymbol{w}'_i$ and $\boldsymbol{w}'_j$ and $D_t(\boldsymbol{w}'_k)$ denotes their depth values.

$$\mathscr{C}_2(i,j) = D_t(\mathbf{w}'_k) < \min(D_t(\mathbf{w}'_i), D_t(\mathbf{w}'_j))\ \forall \mathbf{w}'_k \in \overline{\mathbf{w}'_i\mathbf{w}'_j} \wedge d(\mathbf{w}'_i, \mathbf{w}'_j) > 1. \tag{16.6}$$

Two nodes $[\mathscr{N}_i, \mathscr{N}_j] = [w_i, w_j]$ are then connected by an edge if either criterion $\mathscr{C}_1(i,j)$ or $\mathscr{C}_2(i,j)$ evaluates to true (16.7). For each edge we store its length $|\mathscr{E}_{i,j}|$, which is the Euclidean distance between the two connected nodes.

$$\mathscr{E} = \{[\mathscr{N}_i, \mathscr{N}_j] \in \mathscr{N} \times \mathscr{N} \mid \mathscr{C}_1(i,j) \vee \mathscr{C}_2(i,j)\}. \tag{16.7}$$

Let $g_{\mathbf{r}}(\mathbf{a})$ denote the geodesic distance of an arbitrary point $\mathbf{a} \in \mathbf{W}$ to a given root point $\mathbf{r}$. Using the graph $\mathscr{G}$, we compute the geodesic distances of all point-cloud points using the Dijkstra algorithm [6]:

1. Assign to each node of $\mathscr{G}$ a geodesic distance $g_{\mathbf{r}}(\mathscr{N}_i)$. Set the distance of the root point to 0 and all others to ∞. Label the root point as *open*.
2. From all *open* nodes, find the one with the smallest geodesic distance to the root point, $\mathscr{N}_j = \operatorname{argmin}\ g_{\mathbf{r}}(\mathscr{N}_i)$, and perform the following steps:
   a. Label $\mathscr{N}_j$ as *closed*.
   b. Compute for all *open* nodes $\mathscr{N}_k$ that are connected to $\mathscr{N}_j$ by an edge $\mathscr{E}_{j,k}$ the new geodesic distance $\hat{g}_{\mathbf{r}}(\mathscr{N}_k) = g_{\mathbf{r}}(\mathscr{N}_j) + |\mathscr{E}_{j,k}|$.
   c. If $\hat{g}_{\mathbf{r}}(\mathscr{N}_k) < g_{\mathbf{r}}(\mathscr{N}_k)$, set $g_{\mathbf{r}}(\mathscr{N}_k)$ to $\hat{g}_{\mathbf{r}}(\mathscr{N}_k)$ and mark the current node $\mathscr{N}_j$ as a predecessor of $\mathscr{N}_k$.
   d. If $\mathscr{N}_k$ is not labeled as *closed*, label it as *open*.

The first node to which the geodesic distances are calculated is the node that is closest to the center of the torso. The reason for that is that hands and feet—as the endpoints of the limbs—have high geodesic distances to the torso center and can therefore be found by searching for points of geodesic maxima. This is shown for some selected poses in Fig. 16.4. A very robust way to determine the geodesic maxima positions is to detect at first the node with the highest geodesic distance and connect it to the torso center by a zero weight edge, $|\mathscr{E}_{j,k}| = 0$. After that, the geodesic distances to all nodes are recomputed and the next absolute maximum node

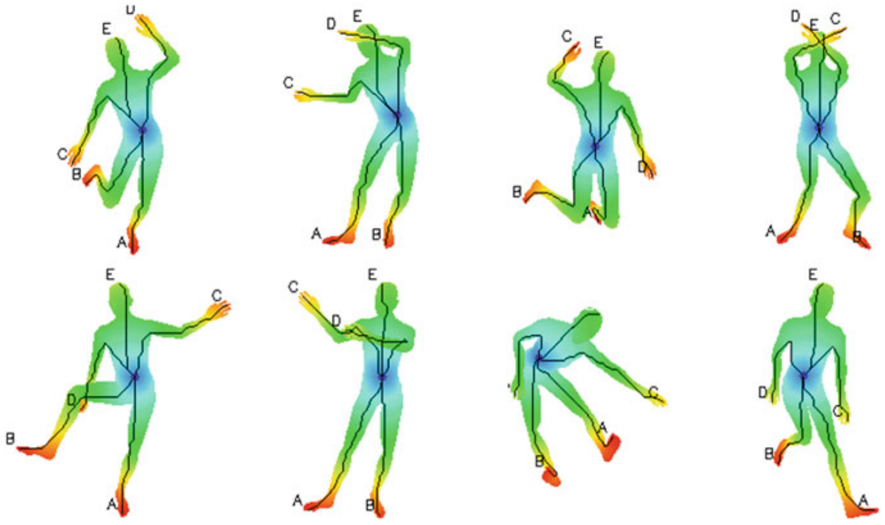


Fig. 16.4 Geodesic distances of all points to the torso center, drawn as colors from *blue* (low distances) to *red* (high distances). *Black lines* are the geodesic paths from the geodesic maxima (labeled as 'A'–'E') to the torso center

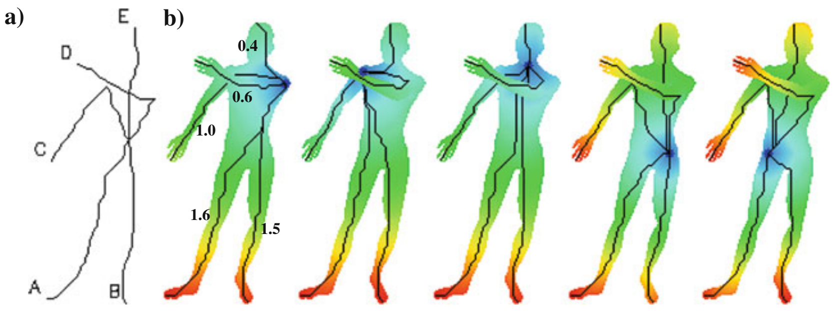
is determined. This is repeated five times in order to find the positions for both hands and feet and the head. In Fig. 16.4 the maxima positions are labeled as 'A'–'E'.

16.3.3 Geodesic Path Assignment

Let $m_i \in M$ denote the detected geodesic maximum points. After they have been determined, the next step is to decide to which limb (extremity) each maximum belongs. There are five extremities, denoted by $\mathscr{L}_1 \ldots \mathscr{L}_5$: the left and right arm, the head, and the two legs. Each extremity has a start point $\{\ell_j\}_{1\ldots5}$, which are the left and right shoulder, the neck and the leg and right hip position, respectively. Having determined the torso model transformation in Sect. 16.3.1, their positions are approximated (see Eq. (16.4)). Geodesic maximum points are assigned to limb start points ℓ_j based on their geodesic distance to ℓ_j. We therefore proceed by computing the geodesic paths between all limb start points ℓ_j and all geodesic maximum points m_i. This is shown for an exemplary pose in Fig. 16.5b. The assignment problem can be considered as a combinatorial optimization problem. A cost matrix $\mathbf{C}$ is created that stores for each geodesic maximum point the path lengths to ℓ_j:

$$\mathbf{C}(i,j) = g_{\ell_\mathrm{j}}(m_i). \tag{16.8}$$

In total, there are $K = |M| \cdot 5$ possible assignments, where 5 is the number of the extremities and $|M|$ the number of detected geodesic maximum points. From all



c)

m_i	LeftArm	RightArm	Head	Left Leg	Right Leg
'A'	1.645	1.602	1.655	1.204	1.077
'B'	1.555	1.609	1.596	1.063	1.113
'C'	1.018	0.689	0.862	1.159	1.100
'D'	0.617	0.868	0.711	0.981	1.060
'E'	0.495	0.486	0.306	0.851	0.886

Fig. 16.5 (**a**) Geodesic maximum points and their paths to the torso center. (**b**) Geodesic paths between maximum points and limb start points ℓ_j (*dark blue dots*). The numbers in the *very left image* are the path lengths. (**c**) Cost matrix used to assign geodesic maximum points to limb start points. The numbers denote the path lengths in meters. Assigned pairs of geodesic maximum points and limb start points are *highlighted*

possible assignments $\Lambda = \{\Lambda_k\}_{1\ldots K}$ between geodesic maximum points and limb start points, we determine the one $\hat{\Lambda}$ with minimal cost, i.e. the one that minimizes the cumulative geodesic distance (Eq. (16.9)).

$$\hat{\Lambda} = \underset{\Lambda_k \in \Lambda}{\operatorname{argmin}} \sum_{\lambda \in \Lambda_k} C(\lambda) \tag{16.9}$$

In Fig. 16.5c the cost matrix **C** for the shown geodesic paths in Fig. 16.5b is depicted. Assigned pairs $\hat{\Lambda}$ of geodesic maximum points and limb start points are highlighted. It can be seen that the geodesic maximum labeled as 'C' is correctly assigned to the right arm, 'D' to the left arm and so on.

16.3.4 Skeleton Fitting

Having assigned the geodesic maximum points to their respective limb points, the final step is to adapt a hierarchical skeleton model in order to determine the final pose. In our approach, each limb is adapted separately. A set of target points U_i to which each limb is adapted needs to be extracted as follows: Let $\{\mathscr{P}_i\}$ denote the set of geodesic paths between the geodesic maximum points and their assigned limb start points $\ell_{\hat{\Lambda}(i)}$ as shown in Fig. 16.6a. A set of target points $\mathbf{U}_i$ is then given by:

$$\mathbf{U}_i = \{\mathbf{w} \in \mathbf{W} \mid \ ||\mathbf{w} - \mathbf{p}_j||_2 < d_e \wedge |g(\mathbf{w}) - g(\mathbf{p}_j)| < d_e, \ \forall \mathbf{p}_j \in \mathscr{P}_i\}. \tag{16.10}$$

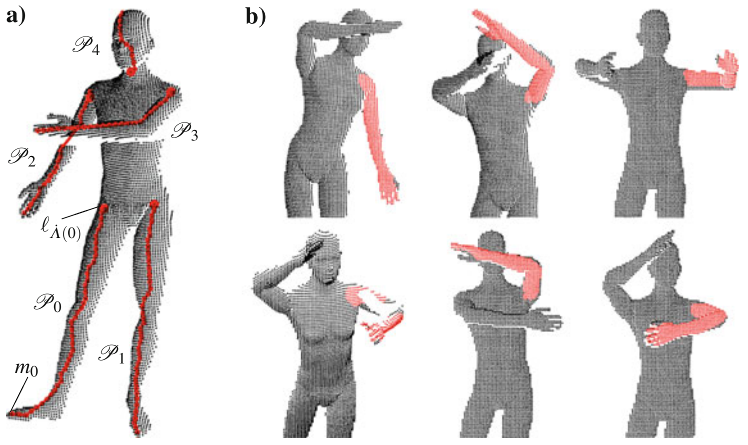


Fig. 16.6 (**a**) Geodesic paths $\mathscr{P}_i$ between the geodesic maximum points m_i and their assigned limb start points $\ell_{\hat{\Lambda}(i)}$. The paths are used to find a set of target points U_i to which a kinematic chain of a hierarchical skeleton model is adapted. (**b**) Extracted sets of target points in various poses to which the left arm of a kinematic skeleton model is adapted (*red dots*)

According to Eq. (16.10), a 3D point $\mathbf{w}$ is considered to be a target point for limb $\mathscr{L}_i$ if its Euclidean distance to a point that is an element of path $\mathscr{P}_i$ is below a threshold d_e and if it has a similar geodesic distance from the torso center. The latter condition prevents points that are close to a path point but belong to a different body part from being added to the set of target points. The threshold value d_e represents the diameter of a limb. We set $d_e = 5\,\text{cm}$ for the arms and $d_e = 10\,\text{cm}$ for the legs and the head. In Fig. 16.6b, a few examples of extracted sets of target points are shown.

In our skeleton model, each limb is represented by a kinematic chain consisting of two joints. The first joint describes the transformation of the upper limb segment, e.g. the upper arm or upper leg, respectively. Accordingly, the second joint describes the transformation of the lower limb segment (fore-arm or lower leg) relative to the first joint. The aim of the model adaptation is to minimize the distance between the target points and the bones of the skeleton model. One way to achieve this is to first determine for each target point whether it belongs to the upper or lower joint and then adapt the joint transformations successively and separately from each other, i.e. to fit, for example, the upper arm joint first and subsequently the lower arm. There are, however, some major problems with this approach:

1. If there are no or there is only an insufficient number of target points for the upper joint, the lower one cannot be adapted correctly, even if target points for it exist.
2. It is not guaranteed that the total error is minimized.
3. The joint rotation cannot be determined in all three axes.

These problems can be addressed when for a given joint also the target points that correspond to its child joints are considered. Therefore, a weight is introduced that determines how much each joint is influenced by each target point. This weight is high for target points that directly correspond to a joint and lower for its children. For each joint, a line segment (bone) to its child joint is computed. Corresponding points are then determined by projecting the target points onto these line segments. The complete algorithm to adapt a kinematic chain consisting of joints $\{J_0, J_1, \ldots, J_n\}$ to a set of target points $\mathbf{U}_i$ is as follows:

1. Let J_j be the first joint of a kinematic chain.
2. Recompute the global positions $\mathbf{b}_j, \mathbf{b}_{j+1}, \ldots$ for J_j and all its children. Also compute the line segments between all joints: $h_j = \mathbf{b}_j + t\mathbf{d}_j = b_j + t(\mathbf{b}_{j+1} - \mathbf{b}_j),\ t \in [0, 1]$.
3. Determine for each target point $\mathbf{u}_k \in \mathbf{U}_i$ and for each line segment h_j the set of corresponding points $\mathbf{A}_j = \{a_k\}_j$:
   a. Project each target point u_k onto each line segment h_j.
   b. If the projected point is an element of the line segment, i.e. $0 < s = (\mathbf{u}_k - \mathbf{b}_j) \cdot \mathbf{d}_j < 1$, add the projected point to the set of corresponding points

$\mathbf{A}_j \leftarrow \{\mathbf{A}_j, \mathbf{b}_j + s * \mathbf{d}_j\}$. Otherwise add the endpoint of the line segment $\mathbf{A}_j \leftarrow \{\mathbf{A}_j, \mathbf{b}_{j+1}\}$.

4. Compute the weights. Determine for each target point $\mathbf{u}_k$ the index of the joint with the smallest distance:

$$\hat{j} = \underset{j}{\operatorname{argmin}} \, ||\mathbf{A}_j(k) - \mathbf{u}_k||_2. \tag{16.11}$$

If $J_{\hat{j}}$ is the current joint J_j, set the weight $\omega_{j,k}$ to 1. If it is a child joint ($\hat{j} > j$), set the weight to a small value ω_{suc}. Otherwise, set $\omega_{j,k} = 0$.

5. Given the weights and the set of corresponding points $\mathbf{A}_j$, solve for the new transformation $\mathbf{T}_j$ of the joint J_j. This is done by minimizing Eq. (16.12) using the quaternion-based closed-form solution provided in [13].

$$\sum_k \omega_{j,k} ||\mathbf{T}_j(\mathbf{A}_j(k) - \mathbf{b}_j) - (\mathbf{u}_k - \mathbf{b}_j)||^2. \tag{16.12}$$

6. Apply the computed transformation $\mathbf{T}_j$ to J_j. Proceed with the next joint $J_j \leftarrow J_{j+1}$ and go to (2).
7. Repeat steps (1)–(6) until the total error (the cumulative distances between the target points and their closest joints) falls below a threshold or a maximum number of iterations is reached.

The actual value of ω_{suc} is not a crucial parameter, but should be much smaller than 1, so that a joint is more influenced by target points that correspond directly to it. We set $\omega_{suc} = 0.05$. In Fig. 16.7 an example of kinematic chain fitting is shown.

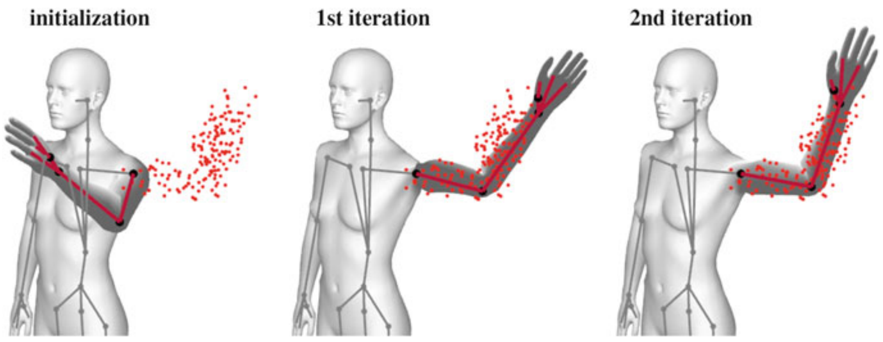


Fig. 16.7 A kinematic chain (*red lines*) that represents the transformation of left arm of a human model is adapted to a set of target points (*red dots*). After two iterations the algorithm has converged

16.4 Gesture Recognition

The gesture recognition process takes as input hand blob to detect features. The features describe an object's underlying characteristics, which should retain distinctly for every specific action. Specifically, the hand features are inherently a static feature set, but for the meaningful gestures these features are measured over a series of frames (*Fr*) to recognize complete gestural patterns (i.e., determined over a period of 1 s) utilizing the Bezier descriptor as follows.

16.4.1 Bezier Descriptors

In the proposed approach, the drawn gestures are recognized by modeling the computed hand centroid points (i.e., blob's center of detected skin segments) using Bezier curves to form Bezier points. The extracted Bezier points are used to compute the features and are then concatenated to form the Bezier descriptors. The proposed approach differs from other approaches as it is not relying on input hand centroid points but on fitted curves to form smoother trajectories for the classification process. Utilizing Bezier curves, the Bezier descriptors contain reliable features even for lower frame rates (i.e., captured frames per second, which is not possible because of its entire dependence on hand features data). The utilization of Bezier descriptors by fitting N (i.e., 5, 10, 15, 20, 30) points makes our approach independent of the HMM state model to be trained (i.e., in HMM, the features length has to be same as the number of states inside HMM, but this features length is hard to handle with varying frame rates). The Bezier descriptor is computed by transforming the hand centroid points which act as control points into a set of Bezier points ($N = 15$) [1, 12]. The Bezier features are calculated by measuring the difference between two consecutive Bezier points which are then binned and concatenated, resulting in a Bezier descriptor.

Mathematically, Bezier descriptors (B_d) are represented for each gesture symbol through the transformation of control points in the form of Bezier points. Normally, polynomials or piecewise polynomials are employed to approximate and represent Bezier curves, B. These polynomials [8, 12] can be of various degrees and are defined as:

$$B(t) = \sum_{i=0}^{d} C_i t^i = C_0 + C_1 t + C_2 t^2 + \cdots + C_d t^d, \tag{16.13}$$

$$C_i = \frac{n!}{(n-i)!} \sum_{j=0}^{i} \frac{(-1)^{(i+j)} P_j}{j!(i-j)!}. \tag{16.14}$$

The representation of these polynomials results in the approximation of Bezier curves [1, 12]. The Bezier curves can be generalized to higher dimensions but it is difficult to exercise control in higher-dimensional space, so we have designed the Bezier points to represent our gestures. A linear Bezier curve [22] which represents a line segment has the following form:

$$B(t) = (1-t)P_0 + tP_1; \quad t \in [0,1]\,, \tag{16.15}$$

where P_0 and P_1 are input points. This mathematical representation [8, 22] is extended to higher-dimensional space as follows:

$$B(t) = (1-t)^d P_0 + dt(1-t)^{(d-1)}P_1 + \cdots + t^d P_d \tag{16.16}$$

$$= \sum_{i=0}^{d} \frac{d!}{i!(d-i)!} t^i (1-t)^{(d-i)} P_i \tag{16.17}$$

$$= \sum_{i=0}^{d} P_i b_{(i,d)}(t), \tag{16.18}$$

where the $b_{(i,d)}$ are Bernstein polynomials and form the basis of space for all the polynomials of *degree* $\leq d$. The motivation of using Bezier curves is to exploit its convex hull property, which ensures that the curve is always confined and controlled with its control points.

Utilizing these Bezier points, the orientation ϑ_i is computed between the two consecutive points P to extract the feature vector. The orientation between the two consecutive Bezier points $(Bz^i_{x,y}, Bz^{i+1}_{x,y})$ is presented as:

$$\vartheta_i = arctan\left(\frac{Bz^{i+1}_y - Bz^i_y}{Bz^{i+1}_x - Bz^i_x}\right); \; i = 1, 2, \ldots, T-1, \tag{16.19}$$

where T represents and length of the gesture drawing path, the $Bz^{i+1}_{x,y}$ and $Bz^i_{x,y}$ are two consecutive Bezier points.

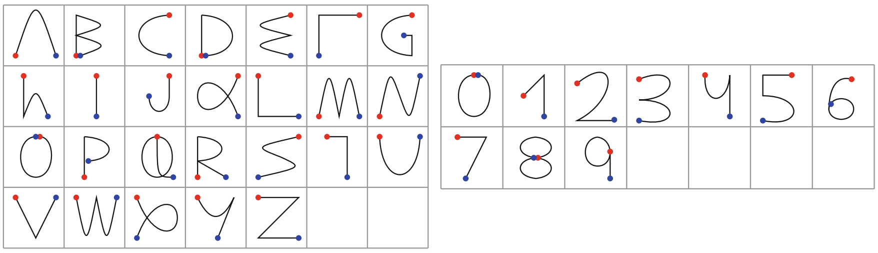


Fig. 16.8 Gesture dataset symbols

16.4.2 Features Binning

In the features binning process, the measured orientation ϑ is binned down into different indexes compatible for the classification process. In the proposed approach, orientation ϑ is scaled down eight bins with the factor of 45° to get quantized features (Qf), and by concatenating these features, Bezier descriptors ($B_d = \{\vartheta_1, \cdots, \vartheta_{T-1}\}$) are formed. Figure 16.9 presents a sequence performed by a subject along with its features in graphs. The images are presented at various instances with hand centroid points (control points in yellow) along with Bezier points (red). The Bezier features ϑ are computed from the consecutive Bezier points, which are binned to generate the feature vector. Here, the gesture stream (1 second data $\approx$ 25 fps) is represented by 14 Bezier points (i.e. shown by green curve in the tracked trajectory). The left graph shows the quantized features generated from

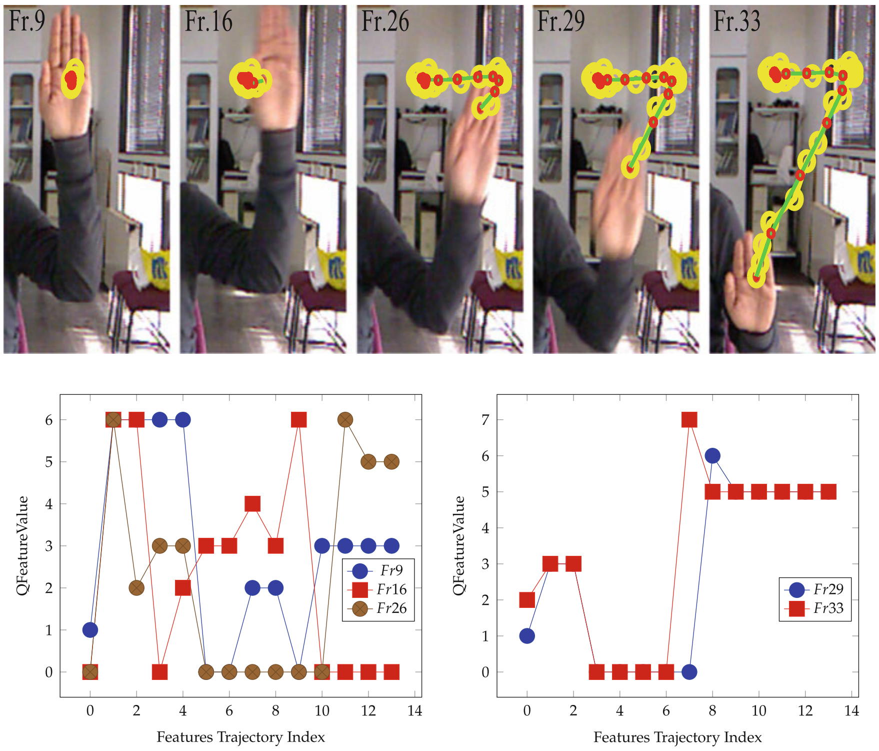


Fig. 16.9 Example of gestural action formation for *Gesture* ‘7’. In the figure, detected hand control points (i.e., *yellow*) and Bezier signature points (i.e., *red*) are presented. The graphs present Bezier descriptors in which the X-axis shows the features vector trajectory and quantized features (Qfs) are shown on the Y-axis. The *left graph* shows the features with no gesture detected *Gesture* ‘-1’ (i.e., Fr 9, Fr 16, Fr 26) whereas the *right graph* presents the features for *Gesture* ‘7’ (i.e., Fr 29, Fr 33)

Bezier points with no gestural pattern detected at *Fr*. 9, *Fr*. 16 and *Fr*. 26, whereas right graph shows Bezier points for detected *Gesture* '7' at *Fr*. 29 and *Fr*. 33.

16.4.3 Gesture Classification

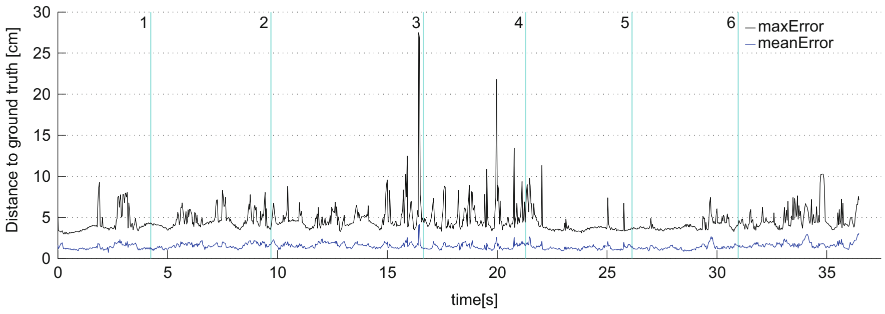
Classification is the final phase, where the input class is assigned to one of predefined classes. The features extracted are the key elements of the classifier because the variability in the features affects the recognition in the same class and between different classes. In gesture classification, the Baum-Welch algorithm is employed to train HMM parameters by discrete vectors (i.e., Bezier descriptors). Inside the HMM, Left-Right banded topology is used with 14 states for classification of gestural symbols, and this is done by selecting the maximal observation probability of gesture model using the Viterbi algorithm.

16.5 Experimental Results

16.5.1 Human Pose Estimation

The proposed method has been validated with a set of synthetic test sequences. Each sequence consists of a series of depth maps that were generated by rendering an animated 3D-character into an OpenGL depth buffer. The character mesh was designed with the MakeHuman application and consists of a 3D mesh and a skeleton (hierarchy of joints). To obtain animation data (sequences of skeleton joint orientations) a human actor performed several motions that were recorded by a Kinect sensor. Even, if the Kinect does not detect the actors' pose correctly, the rendered depth image will match the pose that the Kinect has recognized. In this way, the exact ground truth joint positions are known for each depth image, without having to manually label them. In total, $N = 11{,}250$ frames, separated into ten sequences of variable length between 35 and 45 s and showing various poses, have been created. To each depth image all steps of the processing chain, i.e. measuring geodesic distances, extracting and classifying geodesic paths and adapting a skeleton model to the sets of target points, are applied.

The skeleton model in this approach consists of 16 joints (three for each leg and arm, two within the torso, one for each the neck and the head). Their positions in frame t are denoted by $x_i(t) \in \mathbb{R}^3$ and the ground truth joint positions are denoted by $y_i(t) \in \mathbb{R}^3$. As an error metric, we compute in each frame both the mean Euclidean



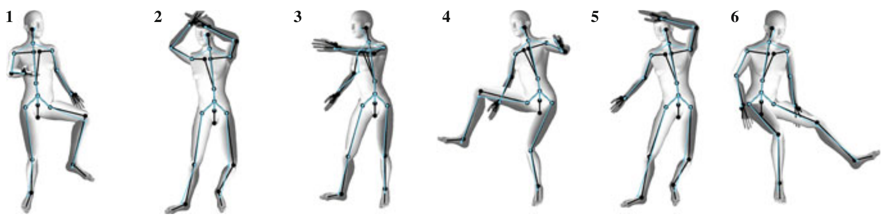


Fig. 16.10 *Top*: Mean joint error (*blue*) and maximum joint error (*black*) per frame in centimeters for a single sequence. *Bottom*: The estimated pose for some selected frames is depicted by the *blue skeleton*. The *black skeleton* (ground truth) was used to animate a 3D-character in order to create synthetic depth images

distance (Eq. (16.20)) and the maximum Euclidean distance (Eq.(16.21)).

$$e_m(t) = \frac{1}{16}\sum_{i=1}^{16} |x_i(t) - y_i(t)|_2 \tag{16.20}$$

$$e_h(t) = \max|x_i(t) - y_i(t)|_2, \ \forall i = 1 \ldots 16 \tag{16.21}$$

$$e_j(i) = \frac{1}{N}\sum_{t=1}^{N} |x_i(t) - y_i(t)|_2 \tag{16.22}$$

Both errors are shown for a single sequence in Fig. 16.10 (top). It can be seen that the mean Euclidean error is about $\overline{e_m}(t) = 2$ cm and therefore very low. The maximum error rises up to 27 cm in single frames. This occurs when the arm is pointing directly toward the camera. In this case, there are too few 3D points to which the skeleton model can be adapted and the relative joint rotations remain unchanged. For several selected frames in Fig. 16.10, the animated characters are shown. It can be seen that also in cases of self-occlusions the estimated pose (blue skeleton) matches closely with the ground truth pose (black skeleton). In Fig. 16.11, we further computed the per joint errors (Eq. (16.22)) averaged over all N generated depth images. The error is between 1 cm for the neck and 4.1 cm for the feet. This can be compared to the also graph-based approach provided in [21], whose authors

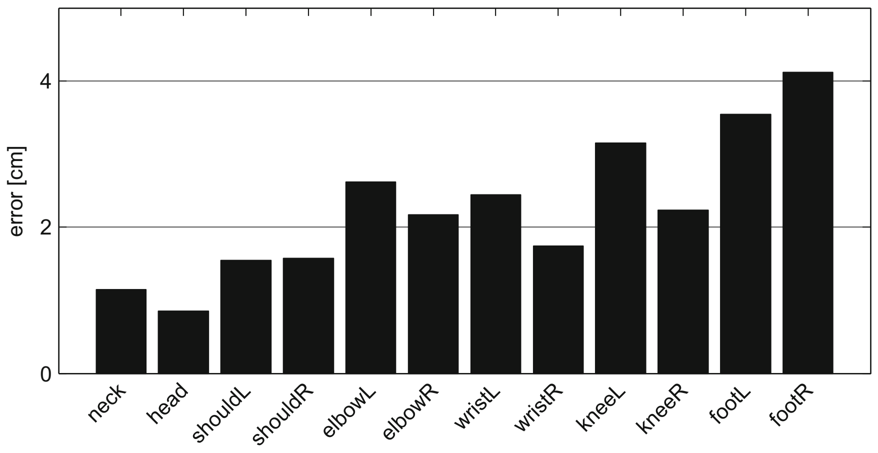


Fig. 16.11 Euclidean distances between estimated and true joint positions averaged over all created depth images

report per-joint errors between 3 and 6.8 cm. The pose is estimated with approx. 40 frames per second. Thus, our method is suitable for the subsequent online gesture recognition system.

16.5.2 Gesture Recognition

The experimental setup involves the tasks of data acquisition, feature extraction and classification. We have demonstrated the applicability of proposed system on real situations where the gestures are recognized as satisfying flexibility and naturalness (i.e., bare hand, complex background). The proposed framework runs with real-time processing at 25 fps on a 2.83 GHz, 4-core Intel processor with 480×640 pixels resolution. The experiments are conducted on 15 video observations per gesture of six subjects performing various hand gestures wearing short to long sleeves, and the gesture dataset contains the symbols from $A \rightarrow Z$ and $0 \rightarrow 9$ (see Fig. 16.8).

Figure 16.12 presents the sequence where the subject is drawing the *Gesture* ‘8’ along with the feature trajectories in graphs. In this sequence, the features are detected from the hand centroid points and transformed to Bezier points $N = 15$ (i.e., marked with red color). The consecutive Bezier points are then utilized to extract Bezier features ϑ, which are then binned to build Bezier descriptors. Bezier descriptors are used inside HMM using LRB topology with 14 states to recognize gestural actions. The left graph shows the Bezier descriptors with gestural action *Gesture* ‘8’ detected at Fr 20, Fr 25, Fr 30, and Fr 33, whereas the right graph shows Bezier descriptors where no gestural action (i.e., Fr 1, Fr 8, Fr 12, Fr 16) is detected.

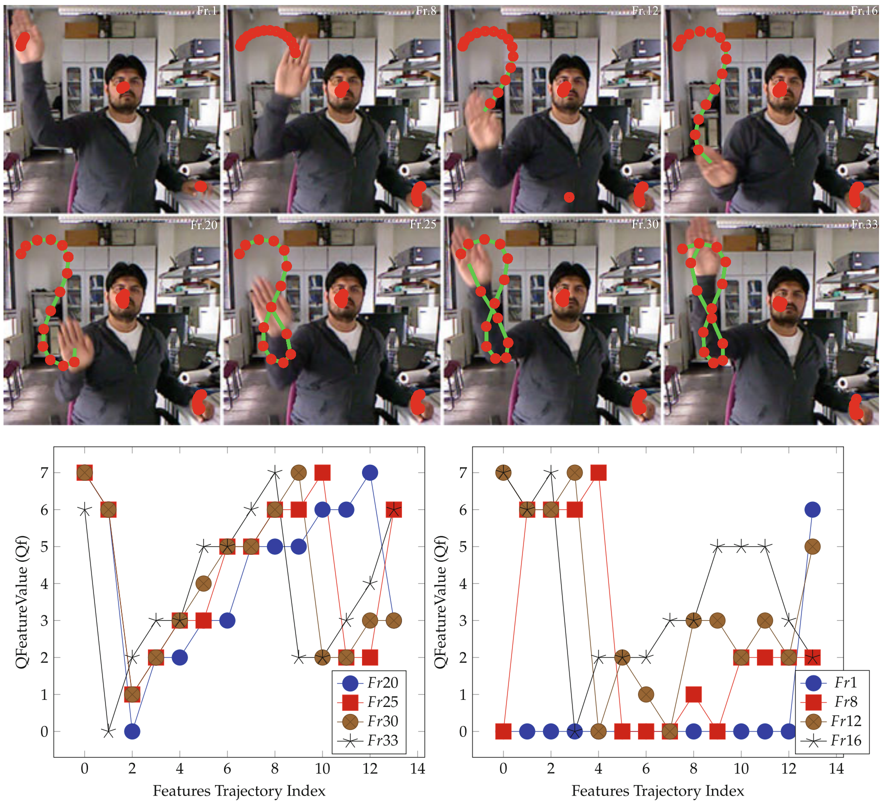


Fig. 16.12 Gestural action of a subject drawing *Gesture* ‘8’ using a Bezier descriptor ($N = 15$). In the sequence, Bezier points (marked as *red points*) are presented whereas the graphs present extracted quantized features with trajectories. The *left graph* shows the features of detected *Gesture* ‘8’ whereas in right graph no gesture *Gesture* ‘-1’ is detected

Figure 16.13 presents the classification results of different Bezier descriptors (i.e., with different HMM states) for the sequence in Fig. 16.12 along with ground truth data. It can be seen that Bezier descriptor $N = 15$ results in the highest recognition rate for this sequence.

The confusion matrix of gestural symbols is presented for $N = 15$ (see Fig. 16.14a), where the diagonal elements show the gestural symbol classification. The higher rate of classification indicates that the Bezier descriptor is capable of recognizing the gestural symbols. Moreover, the performance of Bezier descriptor $N = 15$ is measured against various descriptors like $N = 5$, $N = 10$, $N = 20$ and $N = 30$. Figure 16.14b provides a comparative analysis by adjusting the N parameter (i.e., N is the number of points) for fitting Bezier curves in which the performance of the Bezier descriptor $N = 15$ is superior amongst all with an overall accuracy of 98.3%.

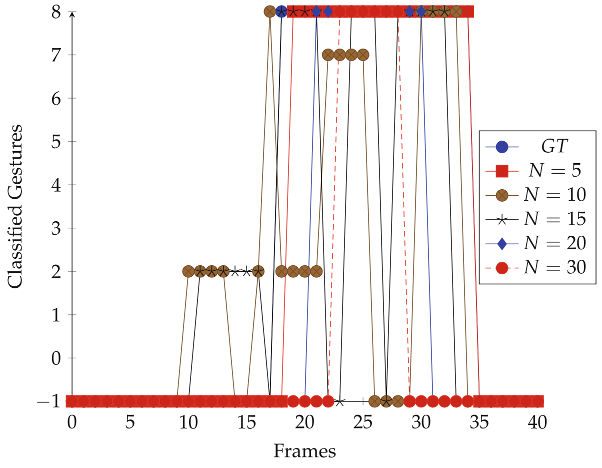


Fig. 16.13 Classification of Bezier descriptors (i.e., $N = \{5, 10, 15, 20, 30\}$) for the sequence using HMM with recognized *Gesture* '8' along with ground truth data *GT*

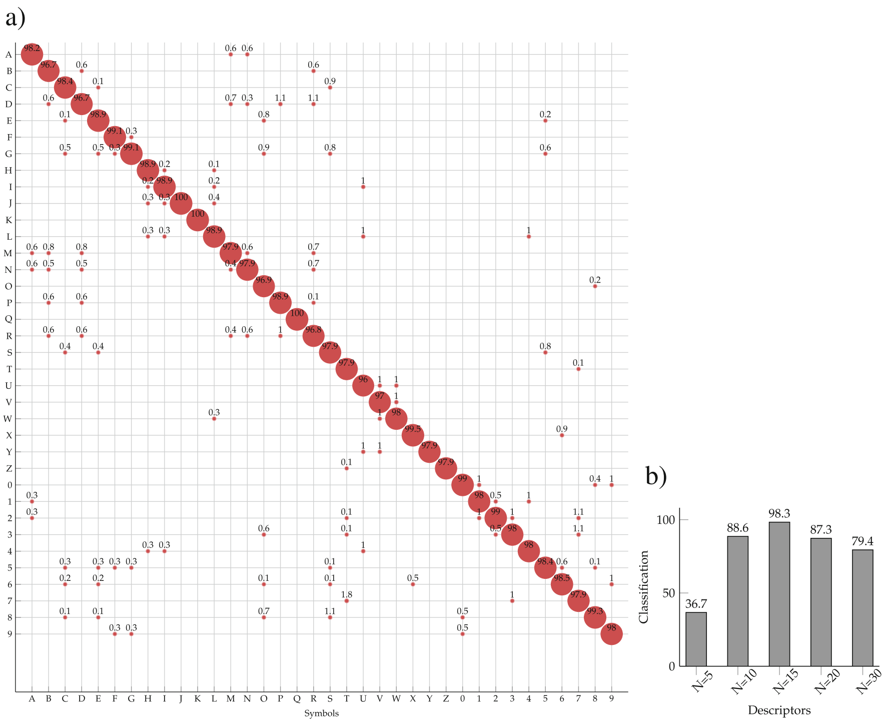


Fig. 16.14 (**a**) Confusion matrix of gesture symbols for Bezier descriptor $N = 15$. (**b**) Classification rate of Bezier descriptors ($N = 5, 10, 15, 20, 30$)

16.6 Discussion and Conclusion

In this work, we present a robust and fast approach for real-time gesture recognition suitable for HCI systems. The basis for the gesture recognition system is formed by an approach for human pose estimation in depth image sequences. The method is a learning-free approach and does not rely on any pre-trained pose classifiers. We can therefore track arbitrary poses as long as the user is not turned away too far from the camera and there is no object between the user and the camera. Based on the measurement of geodesic distances, paths from the torso center to the hands and feet are determined and utilized to extract target points to which a kinematic skeleton is fitted. The approach is able to handle self-occlusions. A database of synthetic depth images was created to compare the estimated joint configurations of a hierarchical skeleton model with ground truth joint positions. On average, the errors were between and 3 and 6.8 cm. In the gesture recognition, features are extracted by modeling the Bezier curves to build Bezier descriptors which are then classified using HMM. Moreover, a comparative analysis of different Bezier descriptors is presented where the Bezier descriptor with $N = 15$ achieves best recognition results. In the future, the dataset will be extended for words and actions to derive meaningful inferences.

Acknowledgements This work was done within the Transregional Collaborative Research Centre SFB/TRR 62 "*Companion*-Technology for Cognitive Technical Systems" funded by the German Research Foundation (DFG).

References

1. Andersson, F.: Bezier and B-spline Technology. Technical Report, Umea Universitet, Sweden (2003)
2. Beh, J., Han, D.K., Durasiwami, R., Ko, H.: Hidden Markov model on a unit hypersphere space for gesture trajectory recognition. Pattern Recogn. Lett. **36**, 144–153 (2014)
3. Chang, J.Y., Nam, S.W.: Fast random-forest-based human pose estimation using a multi-scale and cascade approach. ETRI J. **35**(6), 949–959 (2013)
4. Daniel, C.Y. Chen and Clinton B. Fookes. Labelled silhouettes for human pose estimation. In: International Conference on Information Science, Signal Processing and Their Applications (2010)
5. Dawod, AY., Abdullah, J., Alam, M.J.: Adaptive skin color model for hand segmentation. In: Computer Applications and Industrial Electronics (ICCAIE), pp. 486–489, Dec 2010
6. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numer. Math. **1**(1), 269–271 (1959)
7. Elmezain, M., Al-Hamadi, A., Appenrodt, J., Michaelis, B.: A hidden Markov model-based isolated and meaningful hand gesture recognition. Proc. World Acad. Sci. Eng. Technol. (PWASET) **31**, 394–401 (2008)
8. Farouki, R.T.: The Bernstein polynomial basis: a centennial retrospective. Comput. Aided Geom. Des. **29**(6), 379–419 (2012)
9. Ganapathi, V., Plagemann, C., Koller, D., Thrun, S.: Real time motion capture using a single time-of-flight camera. In: CVPR, pp. 755–762 (2010)

10. Girshick, R., Shotton, J., Kohli, P., Criminisi, A., Fitzgibbon, A.: Efficient regression of general-activity human poses from depth images. In: ICCV, pp. 415–422 (2011)
11. Han, J., Shao, L., Xu, D., Shotton, J.: Enhanced computer vision with microsoft kinect sensor: a review. IEEE Trans. Cybern. **43**(5), 1318–1334 (2013)
12. Hansford, D.: Bezier techniques. In: Farin, G., Hoschek, J., Kim, M. (eds.) Handbook of Computer Aided Geometric Design, pp. 75–109. North-Holland, Amsterdam (2002)
13. Horn, B.K.P.: Closed-form solution of absolute orientation using unit quaternions. J. Opt. Soc. Am. **4**, 629–642 (1987)
14. Liang, Q., MiaoMiao, Z.: Markerless human pose estimation using image features and extremal contour. In: ISPACS, pp. 1–4 (2010)
15. Nanda, H., Fujimura, K.: Visual tracking using depth data. US Patent 7590262, Sept 2009
16. Obdrzalek, S., Kurillo, G., Ofli, F., Bajcsy, R., Seto, E., Jimison, H., Pavel, M.: Accuracy and robustness of kinect pose estimation in the context of coaching of elderly population. In: Engineering in Medicine and Biology Society (EMBC), pp. 1188–1193 (2012)
17. Plagemann, C., Ganapathi, V., Koller, D., Thrun, S.: Real-time identification and localization of body parts from depth images. In: ICRA, pp. 3108–3113 (2010)
18. Qiao, M., Cheng, J., Zhao, W.: Model-based human pose estimation with hierarchical ICP from single depth images. In: Advances in Automation and Robotics. Lecture Notes in Electrical Engineering, vol. 2, pp. 27–35. Springer, Berlin (2012)
19. Raheja, J.L., Chaudhary, A., Singal, K.: Tracking of fingertips and centers of palm using kinect. In: Computational Intelligence, Modelling and Simulation, 248–252 (2011)
20. Rasim, A., Alexander, T.: Hand detection based on skin color segmentation and classification of image local features. Tem J. **2**(2), 150–155 (2013)
21. Rüther, M., Straka, M., Hauswiesner, S., Bischof, H.: Skeletal graph based human pose estimation in real-time. In: Proceedings of the British Machine Vision Conference, pp. 69.1–69.12. BMVA Press, Guildford (2011). http://dx.doi.org/10.5244/C.25.69
22. Salomon, D.: Curves and Surfaces for Computer Graphics. Springer, New York (2006)
23. Schwarz, L.A., Mkhitaryan, A., Mateus, D., Navab, N.: Human skeleton tracking from depth data using geodesic distances and optical flow. Image Vis. Comput. **30**(3), 217–226 (2012)
24. Shotton, J., Girshick, R., Fitzgibbon, A., Sharp, T., Cook, M., Finocchio, M., Moore, R., Kohli, P., Criminisi, A., Kipman, A., Blake, A.: Efficient human pose estimation from single depth images. IEEE Trans. Pattern Anal. Mach. Intell. **35**(12), 2821–2840 (2013)
25. Siddiqui, M., Medioni, G.: Human pose estimation from a single view point, real-time range sensor. In: CVPRW, pp. 1–8, June 2010
26. Van den Bergh, M., Van Gool, L.J.: Combining RGB and ToF cameras for real-time 3d hand gesture interaction. In: WACV, pp. 66–72. IEEE Computer Society, New York (2011)
27. Wang, R.Y., Popović, J.: Real-time hand-tracking with a color glove. ACM Trans. Graph. **28**(3), 63 (2009)
28. Wen, Y., Hu, C., Yu, G., Wang, C.: A robust method of detecting hand gestures using depth sensors. In: IEEE International Workshop on Haptic Audio Visual Environments and Games, pp. 72–77 (2012)
29. Yang, C., Jang, Y., Beh, J., Han, D., Ko, H.: Gesture recognition using depth-based hand tracking for contactless controller application. In: ICCE, pp. 297 –298 (2012)
30. Yeo, H., Lee, B., Lim, H.: Hand tracking and gesture recognition system for human-computer interaction using low-cost hardware. Multimed. Tools Appl. **74**, 1–29 (2013)