Chapter 14
Real-Time Hand Gesture Recognition Using RGB-D Sensor

Yuan Yao, Fan Zhang and Yun Fu

Abstract RGB-D sensor-based gesture recognition is one of the most effective techniques for human–computer interaction (HCI). In this chapter, we propose a new hand motion capture procedure for establishing the real gesture data set. A hand partition scheme is designed for color-based semi-automatic labeling. This method is integrated into a vision-based hand gesture recognition framework for developing desktop applications. We use the Kinect sensor to achieve more reliable and accurate tracking in the desktop environment. Moreover, a hand contour model is proposed to simplify the gesture matching process, which can reduce the computational complexity of gesture matching. This framework allows tracking hand gestures in 3D space and matching gestures with simple contour model and thus supports complex real-time interactions. The experimental evaluations and a real-world demo of hand gesture interaction demonstrate the effectiveness of this framework.

Y. Yao (✉)
School of Mechatronic Engineering and Automation, Shanghai University,
408a HD Building, 99 Shangda Road, Shanghai 200444, China
e-mail: yaoyuan@shu.edu.cn

F. Zhang
Department of Information Science and Electrical Engineering,
Institute of Electronic Circuits and Information Systems, Zhejiang University, Room 603,
Administration Building, 38 Zheda Road, Hangzhou 310027, Zhejiang, China
e-mail: fanzhang@zju.edu.cn

Y. Fu
Department of Electrical and Computer Engineering, College of Engineering, Northeastern University, 403 Dana Research Center, 360 Huntington Avenue, Boston, MA 02115, USA
e-mail: yunfu@ece.neu.edu

Y. Fu
College of Computer and Information Science (Affiliated), Northeastern University, 403 Dana Research Center, 360 Huntington Avenue, Boston, MA 02115, USA

L. Shao et al. (eds.), *Computer Vision and Machine Learning with RGB-D Sensors*,
Advances in Computer Vision and Pattern Recognition,
DOI: 10.1007/978-3-319-08651-4_14

14.1 Introduction

Human–computer interaction (HCI) is an important driving force for computer vision and pattern classification fields. With the development of mobile devices and sensors, hand gestures have become a popular way to interact with tablet PC, smart phones, and personal computers. This trend is not only occurring on the two-dimensional screen, but also happens in the 3D world. However, color images cannot provide enough information for tracking hands in three-dimensional space because much of the spatial position information has to be inferred and this leads to multiple 2D–3D mappings. New sensors, such as Kinect, Xtion, and Leap Motion, can provide the ability to monitor 3D motions, thus make it simple to build systems for human computer interaction via 3D hand movements. This technological progress is very important for applications in the domain of the arts [1], computer gaming [2], computer-aided design [3], and remote control for robots [4]. Combining RGB and depth data will reduce the complexity of target tracking in complicated environments. In addition, the depth information can be utilized to avoid ambiguous mappings between images and hand poses, and generate gestures with clear semantics. In the future, we can expect more devices with built-in depth sensors.

Many hand gesture recognition methods are based on the body of work related to body pose estimation [5]. The state of the art of body estimation techniques began to make use of depth sensors to track human body parts [6, 7]. In recent research, simple pixel features [8] and patches [9] were used as input. They use a random decision forest to recognize different body parts and the orientations of a head, respectively. These methods can be used directly in hand gesture recognition. However, classifiers in these methods must be trained on a large dataset because the recognition process is sensitive to appearance variations of the target shape and backgrounds. Such dataset containing large variations is often hard to achieve.

For those applications using finger movements, accuracy is the primary consideration. It requires the hand to move in a constrained desktop environment and be close to the camera. In environments where the hand is close to the background, segmenting that hand becomes difficult as the background features can be mistaken for the hand and vice versa. In addition, the shape of the hand and the possible hand motions are more complex than those found in the rest of the human body. These problems make it difficult to apply the assumptions made by previous research on body pose estimation.

There are two main challenges in developing hand gesture-based systems at present. The first is how to locate the naked hand and reconstruct the hand pose from raw data. There has been much investigation into hand tracking, hand pose estimation, and gesture recognition. Erol et al. [5] summarized the difficulties faced by these efforts. From the perspective of application development, we summarize the hand tracking, hand pose estimation, and gesture recognition into a single challenge of reconstructing the hand pose from raw data. The second is how to represent the hand model, so that the hand gesture database can be efficiently acquired, and corresponding indexing and searching strategies can be designed to satisfy the real-time hand gesture recognition requirements. Hand models are important for training and recognition accuracy. However, collecting the labeled data required for training is difficult.

We propose a new framework to solve the aforementioned problems. Firstly, we segment a hand into different parts and use a 3D contour model to represent the hand pose configuration. Then, a feature fusion technique [10] is used to unify color and depth information for accurate hand localization. We use a pixel classifier to recognize the different parts of a hand. In order to reduce the workload of establishing real training data, we develop a semi-automatic labeling procedure, which uses both RGB data and depth data to label colored hand patches. Finally, we generate the 3D contour model from the classified pixels. Instead of matching between images, the 3D contour model can be coded into strings. Therefore, the correspondence sample gesture can be found by the nearest neighbor method. Using this framework, we develop a hand gesture-controlled desktop application. Experiments show that gesture matching can speed up efficiently to satisfy real-time recognition requirements.

14.2 Related Work

Over the past decades, many hand gesture-based interaction prototyping systems have been developed. Ali et al. [5] reviewed some of this work. From a technical point of view, the methodologies of pose estimation used in these systems can be roughly divided into model-based generative methods [11], regression-based methods [12], classification-based methods, and examplar-based methods [13].

Most of these methods do not focus on detecting hands. Some of them directly use marker-based motion capture devices, such as a glove fixed with LEDs [14], gloves with colored patterns [15], and data glove [16] to capture the motion of palms and fingers. The accuracy of these systems is determined by the hardware, which is less susceptible to interference from the environment. The shortage is that hardware configurations for these systems are often expensive, inconvenient, and uncomfortable, which make them difficult to use outside the laboratory environment.

People are most likely to adapt to tools for HCI that are less cumbersome. There has been a growing interest in the bare-hand-based gesture-controlled system. Different methods have been developed to build the interactive systems. The essential techniques are varying in these works, but the processing steps are similar, which consists of hand detection and pose estimation. Therefore, we organized them into two main categories, hand tracking and hand pose estimation, and are mainly concerned about the methods that use RGB-D sensors.

14.2.1 Hand Localization and Tracking

Segmenting a hand from a cluttered background and tracking it steadily and robustly is a challenging task. The skin color [17] and background subtraction [18] techniques are the preferred methods for detecting the image regions containing hands. These types of methods, however, make a number of assumptions, e.g., hand is the only active object in the camera scene; otherwise, complex classification methods must be used. Low-level features-based classification methods [19], histogram [20, 21], multi-scale model [22], and motion cues [23] are employed to overcome this problem. Guo et al. [24] presented a method that combines the pixel-based hierarchical feature for AdaBoosting, skin color detection, and codebook foreground detection model to track the dynamic or static hand under changing backgrounds.

In order to improve the robustness and reduce the computation time, current methods combine the ability of a depth camera with RGB information to extract hand regions from multiple candidate regions based on volume bounding box [25], scale [20], pixel probability [26], feature probability [10] and the distance to the camera. Paul et al. [27] have given a comparison on depth image-based hand extraction and RGB image-based hand extraction.

In some relatively early research works, depth data are usually used independently to locate hands. David and Zahoor [28] detected local peaks from low-resolution depth images, which are used as potential hand centers, and a palm radius is used to segment the hand from wrist and arm. Paul et al. [29] used a minimum enclosing ellipsoid to extract the major arm axis. In the perpendicular direction, the local minimum of blob width can be used to segment the hand region from an arm. Depth-based methods are fast. However, the segmentation accuracy is dependent on the method of body pose estimation. Therefore, skin color map is computed to determine which regions of the depth image should be selected [30]. In a recent review article, Han et al. [31] gave a more detailed description on this topic, which covered the depth data preprocessing, object tracking and recognition, human activity analysis, and hand gesture analysis.

14.2.2 Hand Pose Estimation

In hand pose estimation, there are a number of methods developed to find a relationship between 2D hand shape and hand posture [32]. What we called the "hand posture" is commonly defined by an articulated model along with joint angles and the orientation of the palm. These parameters can be assigned different values based on the hand shape extracted from image, and therefore, a large number of postures can be generated. Some of them have predefined semantics, which can serve as gestures and can be further used in human–computer interactive applications.

Model-based methods have also been popular because they can easily incorporate constrains on hand shapes. However, they need complex trackers that are computationally expensive. Due to the fast movement of human hand, image database an indexing technique [23] is employed, which makes it possible to recover hand tracking from each frame. Wang and Popović [15] provided a real-time hand pose estimation system based on this technique.

To remove the ambiguity generated in the 2D projection of hand shapes, depth sensors are used. Mo and Neumann [33] used 3D contours to identify fingers based on a low-resolution depth image. Suryanarayan et al. [34] constructed a volume descriptor in depth space and utilized it to recognize six gestures. Ren et al. [35] used a finger-shape-based distance to distinguish different hand gestures. Liu and Shao [36] proposed a adaptive learning methodology to generate discriminative spatio-temporal features from RGB-D data, which can be used for high-level recognition tasks. A dataset called Sheffield KInect Gesture (SKIG) is provided for hand gesture recognition test.

To reconstruct a full degree of freedom hand model, the different parts of a hand must be pre-labeled and recognized. One of the major approaches for dealing with depth image-based body part recognition is to convert the pose estimation task into a per-pixel classification problem [8]. A simple pixel feature can be used to decrease the computational complexity. This technique can be directly used on hand parts recognition if enough labeled training data are provided. Keskin et al. [37] divided the hand into 21 different parts and used this method to train a per-pixel classifier for segmenting each part. Then, a mean shift algorithm is used to estimate the position of joints. Liang et al. [38] presented a novel framework which exploits both spatial features and the temporal constraints to recover a hand model with 27 degrees of freedom from RGB-D video sequences. So far, only small-scale experimental results on hand pose estimation based on such methods are reported. Different properties of hand poses, such as big deformation and fast motion, make it difficult to identify the different parts of a hand from simple pixel features.

14.3 Framework Overview

As shown in Fig. 14.1, our framework consists of three stages: hand parts classification, hand gesture recognition, and application definition. Each stage contains two different workflows. The upper workflow is used for acquiring training data

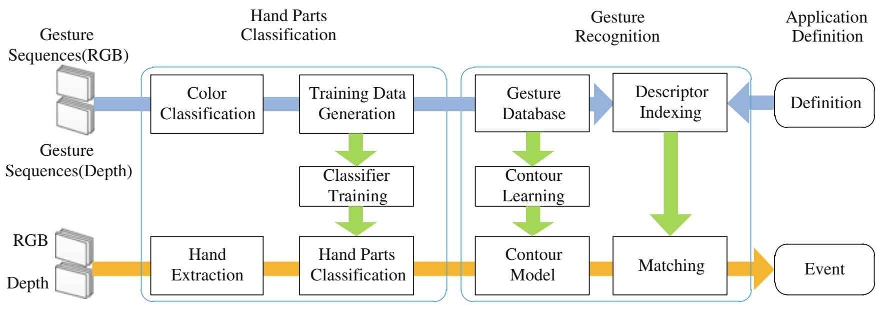


Fig. 14.1 Framework of our hand gesture recognition system

and to define gesture templates; the lower is employed to recognize hand gestures in real-time applications. We decide to use a Kinect camera instead of a dedicated camera as the input sensor for two reasons: (1) the Kinect sensor can provide both RGB and depth information; (2) the accuracy of Kinect is very close to a laser-based devices within a short range [39]. This makes Kinect a good choice for building gesture-driven desktop applications, especially for Augmented Reality applications. In addition, Kinect makes it easier to create a labeled training dataset using a semi-automatic technique that combines the depth and RGB information. Once real hand gestures are captured and labeled in the first stage, we can generate the corresponding contour descriptors in the second stage and provide semantics for them in the third stage. These definitions can then be used in the real applications.

14.3.1 Hand Parts Classification

In hand parts classification stage, we apply the per-pixel technique based on the work by Shotton et al. [8]. To improve the framework's usability in building real HCI applications, we develop a new semi-automatic method for labeling depth pixels containing hand into different classes. A glove with multiple colors is employed to assist the labeling process. The labeling results are fairly noisy and require some manual processing. The output of this procedure is a labeled training dataset. The training set is fed into a classifier based on random decision forest [8]. For real applications, we use a two-step segmentation process: the first step is to segment the hand from the background, which is a binary classification problem; the second is to segment the hand into individual parts based on the per-pixel classifier. Once hands are extracted from a depth image, they are fed into the classifier that roughly partitions them into different parts.

14.3.2 Gesture Recognition

In the second stage, we use an improved 3D contour model based on the model proposed in [10]. The basic idea is that prior knowledge of hand structure can be used to improve the accuracy of classification results. Thus, a contour model is used to recognize both static and dynamic hand gestures.

This stage also contains two workflows. In the upper, training data are converted into contour model-based descriptors and are incorporated into a gesture database. The gesture templates in the database are indexed by a K-d tree structure to speed up the gesture matching procedure. Once the detected hand is segmented into several hand patches, we can generate a 3D hand contour model from it in the lower work flow. This model contains not only gesture descriptors but also the 3D position of the hand. In the demonstration application, we show how they are used for recognizing dynamic gestures.

14.3.3 Application Definition

In order to simplify the process of building real applications, we directly define application-specific gestures in the gesture database. Once a similar hand contour descriptor is matched in the database, an event is triggered to drive the system's response to that gesture.

14.4 Training Data Generation

Training data collection is always a nontrivial task. Inspired by the current research of color-based motion tracking technique [15], we design two configurations of the color glove. As shown in Fig. 14.2, the left pattern includes 9 patches, while the right one includes 14 patches. Both patterns can be used. How to choice depends on requirements of the application. In our previous work [40], the pattern of 14 patches is used. We adopt the one with 9 patches to simplify the recognition in this chapter.

In order to collect the training data in a semi-automatic way, we need to combine both RGB and depth information for labeling. The first task is to calibrate the RGB and depth camera. There are many techniques that can be used to calibrate the two cameras [41–43]. However, in our experiments, we directly obtain the pixels mapping between RGB and depth cameras from the OpenNI Kinect SDK. The second task is to locate the multi-color glove and recognize different color regions. We manually initialize the hand location in the color image and then use mean shift for hand tracking. After the location of the hand is estimated, the different parts of a hand are identified. This is a challenging task because the measured color value can shift from frame to frame due to illumination changes [44], which will result in very noisy signals when depth pixel labeling is performed. In our work, a simple method is employed. The different parts of a hand are initially labeled by using corresponding color mask extracted from the color image. We trained a random forests classifier in the HSV color space and use this classifier to refine the labeling. Finally, we manually finish the labeling work.

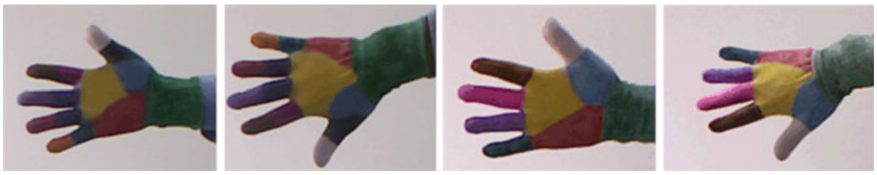


Fig. 14.2 *Color* glove design

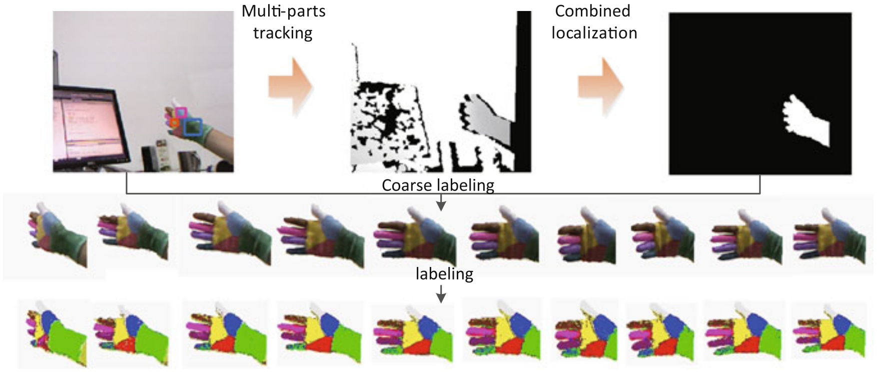


Fig. 14.3 Hand location and labeling

14.4.1 Hand Localization

Depth sensors make segmentation under constrained conditions simpler than ever. However, segmenting the hand from a cluttered background is still a challenging problem, because the shape of the hand is very flexible. In our sample collection stage, both RGB and depth information are used for localization of the hand. Figure 14.3 shows the procedure of recognizing the hand.

First, we ask users manually select several parts within a hand region (top left image in Fig. 14.3). This selection uses a fixed sequence, which usually includes three color regions from the first image of a gesture sequence. Then, a mean shift method is used to track multiple hand parts simultaneously for these gesture sequences. Therefore, the location and direction of the hand can be confirmed during the tracking process.

At the same time, the shapes of foreground targets are segmented from the depth image. We make use of the relationship between RGB and depth image to find where the hand is located. This process generates a hand mask. Each depth pixel in this region can be assigned a RGB color, which will generate a coarsely labeled gesture sequence on depth pixels.

14.4.2 Color Classification

Coarsely labeled results need to be further refined. As shown in the last row of Fig. 14.3, we use a random forests classifier to generate labels depending on pixel colors on the glove. Those colors are converted into HSV space. We trained a random forests classifier based on the 9 groups of manually labeled samples. In the training and classification process, only the component of hue is considered.

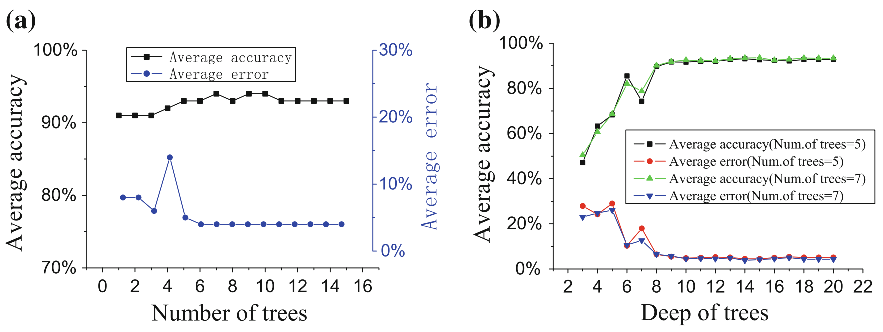


Fig. 14.4 **a** Average accuracy and average error vs Number of trees **b** Average accuracy and average error vs Deep of trees

Figure 14.4 shows the performance of the random forests-based labeling. In the experiments, 50 % manually labeled samples are used as training set, and the other data are used to test. Figure 14.4a reports the trends of average recognition accuracy and average false identification rate with different number of decision trees. Figure 14.4b gives the average recognition accuracy and average false identification rate with different deep of the random forests trees.

The output of the color classification step is used as training data for generating 3D hand contours for the gesture database. During the semi-automatic labeling process, there are many incorrectly labeled pixels. After color classification, manually labeling process is still employed to remove this noise. There are 2,400 labeled frames of hand samples from 6 people in our training database. Five different types of hand gesture sequences are included in the database. By using the classification results, the average labeling time is decreased from 6 min to 30 s for each image in our experiments.

14.5 Hand Patches Segmentation

There are two problems that need to be solved in the hand pose estimating process: hand extraction and hand parts classification. Without strong assumptions about the range of activities, segmenting hands from a cluttered background is difficult, because of the interference of different objects in the scene and other body parts, such as arms, head, and shoulder. Segmenting a deformable and rotatable hand into different parts is also a challenging task. We use a two-step segmentation process in our approach. One is the full hand extraction step, and the other is hand parts segmentation. The former step uses a classification and tracking procedure to distinguish hand objects from other objects. The latter step is to segment hand into parts depending on the feature extracted from depth image.

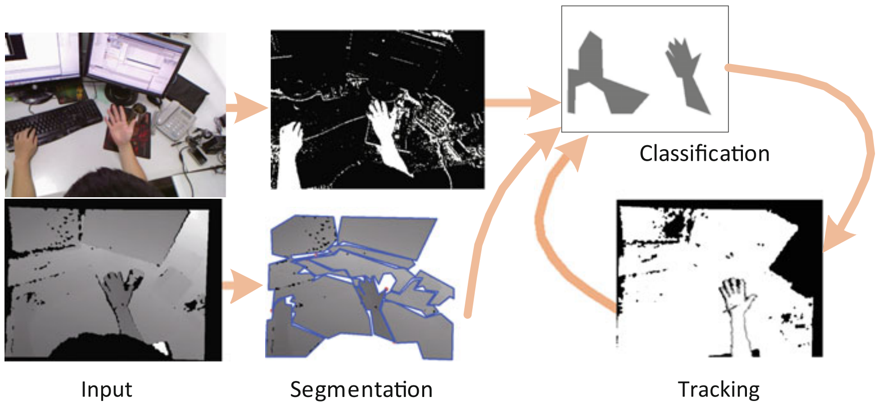


Fig. 14.5 Hands extraction

14.5.1 Hand Extraction

In order to discriminate hand-like objects from other foreground objects, a tracking technique is used. The complete procedure is shown in Fig. 14.5.

For RGB images, the skin color pixels are extracted from RGB image by [45] to generate the skin color mask (as shown in Fig. 14.5 upper middle).

For the depth image, we first assume that hands move within a certain distance range to the camera (distance <1.6 m). Outside this range, part of the depth data is often missing due to multiple reflections and scattering on the surface of the hand. The assumption is also helpful in removing some of the noise. Then, a dynamic threshold segmentation is used to extract multiple targets from the depth data (as shown in Fig. 14.5 bottom middle). In this segmentation process, we convert the depth image into a 8-bit gray image. A histogram with 256 bins is constructed on this image, which is further smoothed by a Gaussian kernel. The local max and min values in the histogram can be found by computing the gaussian derivatives for each bin. Therefore, we can segment the depth image into multiple regions by the depth clusters.

Both depth and RGB images are used to get the initial segmentation. The candidate targets containing enough skin-colored pixels are kept as potential hand locations. In experiments, this percentage is set to a constant value. We found that areas containing 70 % skin-colored pixels perform well during classification. A simple 2D shape feature is used to classify targets into hands and other objects. After this process, the selected targets are fed into a Kalman filter-based tracker to avoid any jitter, noise, occlusions, and large deformations of the hand in the input video. We use a standard Kalman filter described in [17] to track the trajectory of the palm center in 2D space. The segmentation and classification procedure are done for each frame. Even if the hand is lost in a certain frame due to noise or temporary occlusion, it will be recovered in the successive frames.

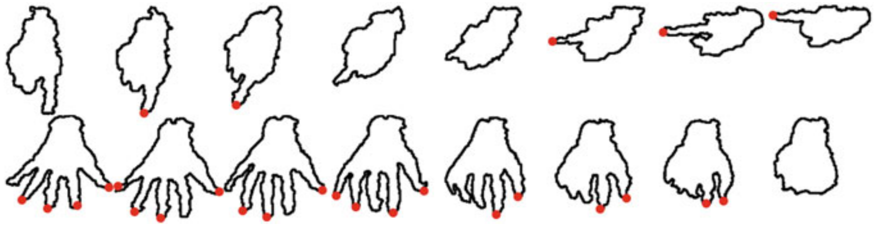


Fig. 14.6 2D shape features defined on hand contour. The *red* points indicate the locations of features

We use a 2D silhouette feature for determining if an object is a hand or not. This feature uses local curvature on the contour to describe the geometry of the fingertips. For each pixel on a contour, the feature is calculated by

$$\mathrm{acos}((\mathrm{x}_{i-1}^{A} - \mathrm{x}_{i}^{A,H}) \cdot (\mathrm{x}_{i+1}^{A} - \mathrm{x}_{i}^{A,H})) < T, \tag{14.1}$$

where x^{A} represents the coordinate of a depth pixel on a target contour A. A is an approximated curve computed from the original hand contour extracted from a depth image. $x_i^{A,H}$ is the coordinate of a depth pixel on A's convex hull H. T is an empirical threshold. As shown in Fig. 14.6a, the red points indicate where the features are located. We found that $T = 0.8$ performs well in practice. It is not very robust because the shape feature cannot always be detected from the contour in some frames. In order to segment the hand from other extracted targets, we learn the prior probability of the number of features from a group of training depth images.

Bayes rule is used to compute the posterior probability of an object being a hand. The posterior is represented by $P(h = 1|\mathscr{S})$, which represents the likelihood of observing a hand object when shape features $\mathscr{S}$ are given, we have

$$P(h = 1|\mathscr{S}) = \frac{P(\mathscr{S}|h = 1)P(h = 1)}{P(\mathscr{S})}, \tag{14.2}$$

where $P(h)$ is the prior probability that measures whether the target is a hand $(h = 1)$ or not $(h = 0)$ without observing its color. We can estimate the probability density function of $P(\mathscr{S}|h = 1)$ and $P(\mathscr{S}|h = 0)$ from the training database. Here, $P(\mathscr{S}) = P(\mathscr{S}|h = 1)P(h = 1) + P(\mathscr{S}|h = 0)P(h = 0)$. Candidate regions are chosen if their posterior probability is greater than a threshold.

14.5.2 Hand Parts Classification

Depending on a weak spatial prediction, the position feature defined in the depth image [8] is simple and effective. However, it is not rotation invariant. The property of rotation invariance is important for hand pose estimation, especially in the case, the support of the body pose estimation is missing.

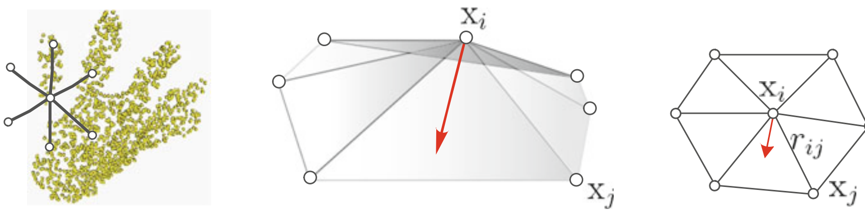


Fig. 14.7 Position feature is a geometrical feature defined on the depth image (*left*), which represents local details. The *red arrow's* direction approximates the normal, and the size approximates the mean curvature (*middle*). In the 2D depth image, this feature can be computed by a neighborhood sampling descriptor (*right*)

In order to overcome this problem, we create a new feature, which is defined over the 3D surface constructed by the depth pixels. The distribution of depth values in a neighborhood of a pixel is considered to identify which part of the hand that pixel belongs to. As shown in Fig. 14.7(left), the relationship of depth pixels in a neighbor domain can be represented as a graph $G = (X, E)$, with pixels X and topology E. Where $X = [\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_n]$, $x_i = \{x_{iu}, y_{iv}, d_I(\mathrm{x}_i)\}$ is the depth pixels set. $d_I(\mathrm{x}_i)$ is the depth value of pixel x_i. Inspired by [46], we define the position feature for each pixel x_i by

$$f_i(I, x) = \sum_{i,j \in E} \omega_{i,j} \left(x_j - x_i\right), \tag{14.3}$$

where $\sum_{i,j \in E} \omega_{i,j} = 1$.

A graphical description of this position feature is given by Fig. 14.7(middle). The neighbor domain information is introduced by the topology. In Eq. (14.3), the direction of $f_i(I, \mathrm{x})$ approximates the pixel's normal, and the size of $f_i(I, \mathrm{x})$ represents the mean curvature. We only use the size; thus, the position feature is rotation invariant. In order to improve the computational efficiency, we use the pixel's normal and the mean curvature to compute the feature. In the algorithm, the key parameter is the feature scale parameter $r_{i,j}$, which represents the distance between $\mathrm{x_i}$ and $\mathrm{x_j}$ (Fig. 14.7 right). By using a constant $r_{i,j}$, we can define the position feature on the depth image as

$$\delta_i = L(\mathrm{x}_i) = d_I(\mathrm{x}_i) - \frac{1}{n} \sum_{j \in E} s_I(\mathrm{x}_j), \tag{14.4}$$

where $s_I(x)$ is the depth value of pixel x, n is a predefined number of sampling points, and

$$s_I(\mathrm{x}) = \begin{cases} d_I(\mathrm{x}) & \mathrm{x} \in \text{hand} \\ b & \mathrm{x} \notin \text{hand} \end{cases}. \tag{14.5}$$

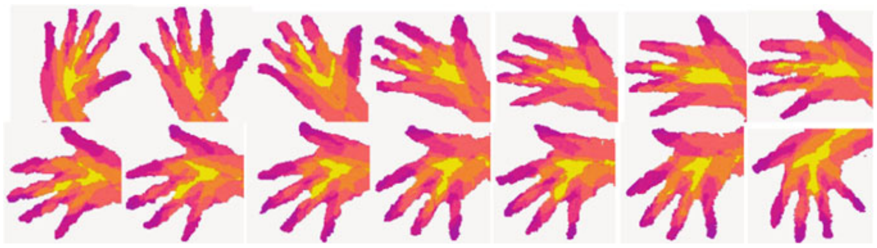


Fig. 14.8 Position feature in a rotated hand

Figure 14.8 shows that examples of the position features are computed on a series of rotated hands. For illustration purposes, feature values are mapped into the RGB color space. As can be seen in Fig. 14.6b, there is no clear boundary between the palm center and hand edge. This means that the signal is weak. To overcome this problem, we use a random decision forest to combine these features in all training data. Since the contour of each hand is already detected, the position feature does not depend on a specific background.

We use the 2D shape feature and the position feature to segment hand patches. The geometry feature that was used in the original framework [10] is removed, because its contribution for improving the accuracy is insignificant. Figure 14.8 shows the result of classifying different parts of a hand using features of different scale extracted from 1906 frames of test data. To improve the accuracy of classification and reduce the false-positive rate, we refine the estimation results with a contour model in the "gesture recognition" stage.

14.6 Gesture Recognition

For many applications, the goal of hand gesture recognition is not to recognize arbitrary 3D hand pose, but to recognize if a gesture is one of a fixed number of hand poses. Therefore, in this work, we use a database indexing technique to implement gesture recognition. A hand contour database is collected by using the training samples. In the following sections, we give the description of the contour model and the similarity measures.

14.6.1 Contour Model

The final output of the estimation process is a labeled 3D contour. There are several advantages of using labeled 3D contours as hand models. First, a contour is a simple structure. It is easier to match two contours with different scales than matching image patches or articulated models. Second, the representation of a contour only needs a small size descriptor, so it is more appropriate for database indexing-based gesture recognition techniques, in which a large number of samples are collected for gesture matching. Third, it is convenient to convert contours to other models, such as articulated model and 3D skin model.

Contour Descriptor. The hand contour model C is represented by a ordered list of points on the contour $\mathbf{c} = \{\mathbf{v}_1^c, \mathbf{v}_2^c, \dots \mathbf{v}_n^c\}$ with its corresponding label vector $\mathbf{l} = \{l_1, l_2, \dots l_n\}$. Where v is a vertex on contour c. The value of n is determined by the distance of the hand to the camera and describes the size of a hand contour and is decided by the distance of the hand to the camera. Label vector provides the information about specific hand contour segments belonging to certain hand parts. As a descriptor, both 3D coordinate sequence and labeling index sequence cannot provide a rotation and scale invariance. We add another sequence, $\mathbf{m} = \{m_1, m_2, \dots m_n\}$, for the contour representation. $\mathbf{m}$ is a normalized length sequence.

$$m_i = \frac{\text{Length}(\{\mathbf{v}_j^c | l_j\})}{\text{Length}(\mathbf{c})}, \tag{14.6}$$

where j represents the continuous indexing. The descriptor of the ith hand contour C^i is represented by the $\{l_1^i m_1^i, l_2^i m_2^i, \dots l_d^i m_d^i\}$, where d is the dimension of the descriptor. By selecting d, most significant m_i, d can be set to a fixed dimension.

14.6.2 Contour Generation

Given a 3D contour $\mathbf{c}$, a natural method for constructing $\mathbf{l}$ is to directly search the set of labeled points and, for each point v_i, find the class with the maximum probability. We define the probability as

$$P(\mathbf{c}|\mathbf{v}_i^c) = \frac{1}{n}\gamma_i \sum_{j \in N(i)} P_j(\mathbf{c}|\mathbf{v}_j), \tag{14.7}$$

where $N(i)$ represents a vertex set in the neighbor area of $\mathbf{v}_i^c$. $P_j(\mathbf{c}|\mathbf{v}_j)$ can be deduced from $P(\mathbf{c}|\text{x})$. γ_i is a learning parameter defined as

$$\gamma_i = \frac{p_c^i}{M} \sum_{k \in \varsigma} P(\mathbf{c}|\mathbf{v}_k^c), \tag{14.8}$$

which considers the continuity of the 3D contour. Here, ς is a contour segment that consists of some subset of indices of $\mathbf{v}_i$ and its neighbor domain. M represents the size of ς. p_c^i is a learning parameter, which defines a connectivity relationship between different contour segments. By using this model, the gesture recognition

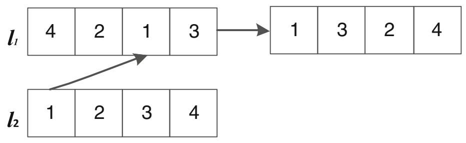


Fig. 14.9 Contour descriptor alignment

process can be described as a sequence matching problem, which can also be used to recover the 3D hand position.

14.6.3 Contour Matching

In order to measure the similarity of two contours, our task is to find a distance function $D(\cdot)$ for fast matching. Once $D(\cdot)$ is determined, the descriptor can be aligned according to Fig. 14.9. That means given a contour descriptor C^i, the query result from a database should satisfy

$$C^i = \mathop{\arg\min}_{i\in\Psi, j\in\Omega} D(C^i, C^j)w, \tag{14.9}$$

where Ψ is the test set of contours, Ω represents the template database. w is a weight, which is computed by

$$w = \sum_{k=\{1,\cdots d\}} l_k^i \ \mathtt{xor}\ l_k^j. \tag{14.10}$$

There are two possible cases where errors arise in generating of hand contours: whole-to-whole matching and whole-to-part matching. We use Smith–Waterman [47] algorithm to deal with both cases. In our application, we select the arm class as the start of the descriptor, because in the data, the arms are typically longer than any other hand parts and are always visible.

14.7 Validation

We evaluated our method on the dataset collected in Sect. 14.4. A total of 2564 frames was captured, each of which includes calibrated depth data, labeled hand parts information, and a mask image of the hand region. All these data are compressed and stored in a SQLite database. In the hand gesture recognition test, we compared our method on a second RGB-D dataset provided in [35].

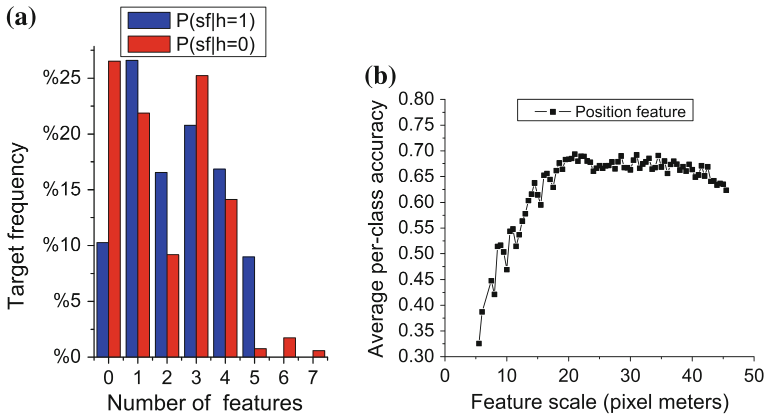


Fig. 14.10 **a** Prior distribution of shape features. **b** Feature scale and average per-classes accuracy

14.7.1 Hand Detection

In most indoor bare-hand interaction applications, hands and objects often appear together. So we assume $P(h = 1) = P(h = 0) = 0.5$; thus, Eq. (14.2) can be simplified to

$$P(h = 1|\mathscr{S}) = \frac{P(\mathscr{S}|h = 1)}{P(\mathscr{S}|h = 1) + P(\mathscr{S}|h = 0))}. \tag{14.11}$$

The distribution of $P(\mathscr{S}|h = 1)$ is learned from 600 frames that are randomly chosen from the training database, while other objects of size similar to the hand, such as head, bottles, apples, keyboards, computer mice, and books in the database, are used to estimate $P(\mathscr{S}|h = 0)$. These two distributions are given in Fig. 14.10a, where the blue bars and red bars represent the statistic results on hand and other objects, respectively. With depth information, shape features can be extracted quickly employing a pre-defined threshold. The number of shape features is counted and used to compute Eq. (14.2). This method is tested on the remaining 1,800 images containing a cluttered background. The hand detection procedure achieves 54.5 % accuracy. After using the skin color segmentation described in section V, the accuracy increases to 94.7 %. The classification results are heavily dependent on the environment. In our application, the frame-loss rate is less than 1 % during the tracking procedure. Multiple hands can be processed simultaneously with small amount of additional calculation. The disadvantage is that it will not work in the following scenarios: (1) the scene contains the body; (2) one hand is occluded by another hand; (3) a person is wearing gloves. Body skeletal trackers can be used to solve these problems.

14.7.2 Hand Parts Classification

In hand parts classification experiment, we select 600 frames from 6 subjects for training and use the other frames for testing. The training samples and testing samples are collected from multiple indoor environment and with different backgrounds and camera view angles.

Figure 14.10b shows the relationship between position feature scale and per-part recognition accuracy rate. The mean per-class accuracy is used as a baseline. Instead of using a single shape feature, we use the pixel's normal and the pixel's mean curvature defined in Sect. 14.5 as the position feature. Compared with the per-part recognition accuracy rate in our previous work [10], the classification results show a significant improvement. This curve will change slightly when different topologies are selected for shape feature.

Table 14.1 shows the per-pixel classification results of detecting different hand patches. The experiment is done on 1,800 frames, where "CMC" indicates the region near the root of little finger, and "TM" indicates the region near the root of thumb. The average accuracy of recognition is 58.48 %. However, the misclassification rate is still high. Shotton et al. [8] showed that using more training samples will improve the mean per-class accuracy. However, labeling real data and generating synthetic training data are nontrivial. Combining more features into the position feature will also improve the mean per-class accuracy [10] but too many features might result in the over-fitting problem.

Figure 14.11 shows the per-class accuracy for pixels located on the hand contour (Fig. 14.11 black) and the per-class accuracy by using the contour model (Fig. 14.11 red). Compared with the results provided by the pixel classification, there is a large improvement in contour model-based classification. The average accuracy of hand parts classification is improved from 58.48 % to 77.68 %. The error rate is still large, but hand orientation can be estimated from the contour, which can also be used in gesture matching.

Table 14.1 Per-pixel hand parts classification results

Category	Sample quantity	True positive (%)	False positive (%)
Thumb	1,574,265	51.0	46.30
Index	1,335,410	33.5	3.10
Middle	1,540,891	74.3	80.60
Ring	1,383,038	66.3	57.40
Small	1,085,247	29.6	21.60
CMC	2,756,895	63.0	44.80
Palm	2,587,585	85.1	83.20
TM	1,602,793	29.8	3.94
ARM	9,866,422	93.7	7.96
Average accuracy		58.48	38.78

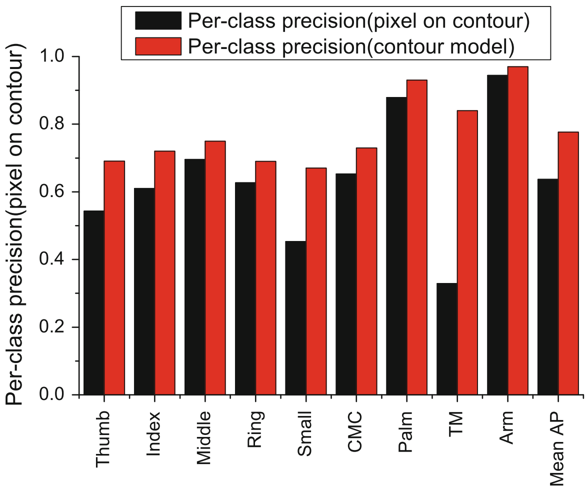
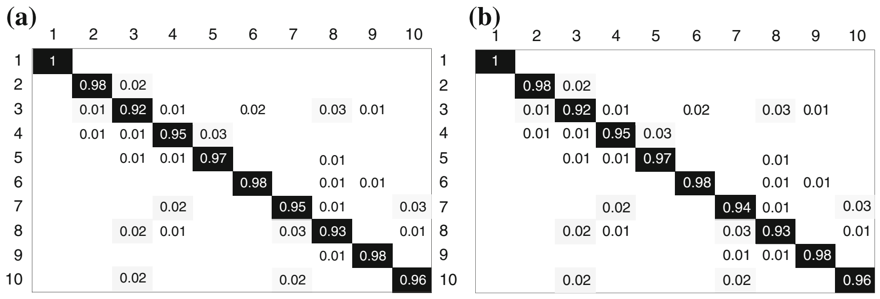


Fig. 14.11 Per-pixel classification accuracy and per-segment classification accuracy with contour model

14.7.3 Gestures Recognition

Our method is tested on the database provided by the authors of [35] without changing any parameters and features. Because these samples do not provide the hand parts segmentation, we capture the same gestures and manually labeled them for each subject. Then, the entire dataset of [35] is used for testing. Figure 14.12 shows the test results in confusion matrix. Compared with the near-convex decomposition-based Finger-Earth Mover's Distance (FEMD) method, our method achieves a similar accuracy by using 40 training samples. Although the recognition rate of some categories has not been significantly improved, the average true-positive rate is better.

We use a desktop computer with an AMD Phenom II X4, 3.0 GHz CPU with 3G RAM to run the test. For each frame, the computation time includes approximately 10 ms for target segmentation, 22 ms for tracking, 17 ms for sampling and features computation, 3 ms for classification, and 32 ms for contour matching. We use a small gesture database that includes 320 hand contour descriptors to test the matching of hand contours. Our C++ implementation processes each frame in 87 ms. GPU acceleration is not used. It is capable to do a real-time gesture recognition on a faster machine. A GPU version is needed when the application contains complex interactions and scene rendering.



(a)

	1	2	3	4	5	6	7	8	9	10
1	1									
2		0.98	0.02							
3		0.01	0.92	0.01		0.02		0.03	0.01	
4		0.01	0.01	0.95	0.03					
5			0.01	0.01	0.97			0.01		
6						0.98		0.01	0.01	
7				0.02			0.95	0.01		0.03
8			0.02	0.01			0.03	0.93		0.01
9								0.01	0.98	
10			0.02				0.02			0.96

(b)

	1	2	3	4	5	6	7	8	9	10
1	1									
2		0.98	0.02							
3		0.01	0.92	0.01		0.02		0.03	0.01	
4		0.01	0.01	0.95	0.03					
5			0.01	0.01	0.97			0.01		
6						0.98		0.01	0.01	
7				0.02			0.94	0.01		0.03
8			0.02	0.01			0.03	0.93		0.01
9							0.01	0.01	0.98	
10			0.02				0.02			0.96

Fig. 14.12 Confusion matrix of gestures recognition results. The gesture categories is ordered by the original database. **a** Near-convex decomposition + FEMD [35]. **b** Hand contour matching

14.8 System Application

In order to evaluate the feasibility and effectiveness of the proposed framework, we perform two case studies.

14.8.1 Augmented Reality Molecular Assembler

The fist application is based on a HCI program named AR Chemical [48], which is an application that utilizes a tangible user interface for organic chemistry education. The interface uses an Augmented Reality (AR) mouse and AR makers to allow users to compose and directly interact with 3D molecular models. All interactions are done on a real desk. The operation includes moving, dragging, positioning, and assembling virtual structures in a 3D space. Therefore, delicate and complex interactive behaviors and hand position information are needed.

We simulate this system and provide three hand gestures to replace the original operation that used with the AR paddle and computer mouse. First, we use keyboard to select atom and then use raising thumbs to simulate the mouse button. The finger pointing action is used for positioning objects.

Once a gesture is detected, we need to record the starting position of the hand and rotate the indicator mark on the 3D molecule, which inserts the atom into the right position pointed to by the finger. Finally, the waving hand with a different orientation is used to assist this assembling process by rotating the assembled molecule on the platform.

The dynamic gesture recognition is realized in the application layer. We set up two gesture buffers in the application: one is for reserving sequential contour with 3D vertices to get hand position; the other is for holding hand contour descriptors to match the hand pose. We define simple interaction action rules on the buffers. This scheme is easy to extend.

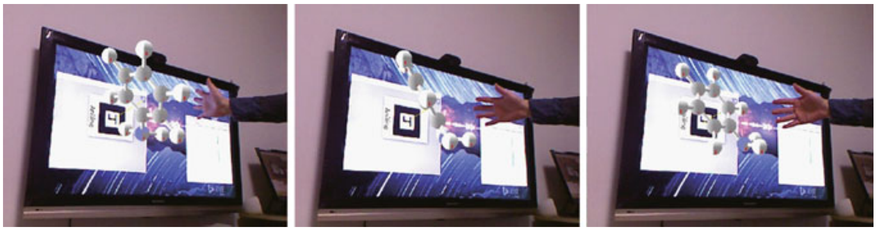


Fig. 14.13 Waving gesture for rotation control

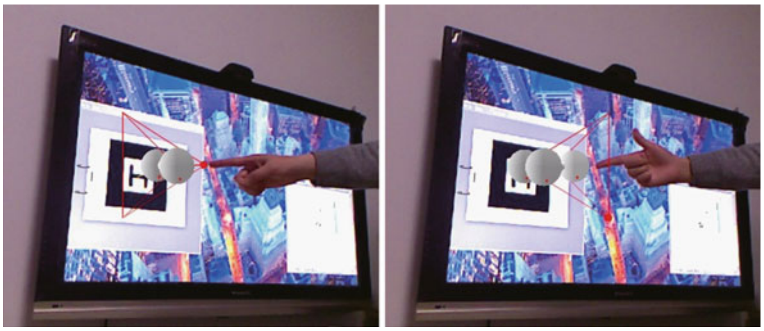


Fig. 14.14 Atom position selection and molecular assembling operation

Figure 14.13 shows the rotation operation. Figure 14.14 is sampled from a procedure of adding an atom into a molecule. This combination of four hand gestures can drive the real-time visual interaction with complex visual feedback. Such type of interactions can increase natural feeling of the operation. The Kinect-based system is able to overcome the problem of illumination changing. We believe that other applications, such as digital sculpture, painting and computer-aided design system could benefit from this framework as well.

14.8.2 Xerrys Intelligent Rehabilitation System

By combining the body pose estimation, we applied our technology to a medical system called Xerrys Intelligent Rehabilitation System for Hospitals, in short XIRSH. XIRSH is a platform of new Kinect-based applications. The major objective of XIRSH is to help patients conduct trainings and practices at the late stage of rehabilitation in hospitals. It can be considered as a comprehensive rehabilitation tool, which provides new methods for occupational and physical therapy and combines the motion control with cognitive and speech rehabilitation. A variant version of XIRSH can be served as a home based application to further extend the tele-rehabilitation process.

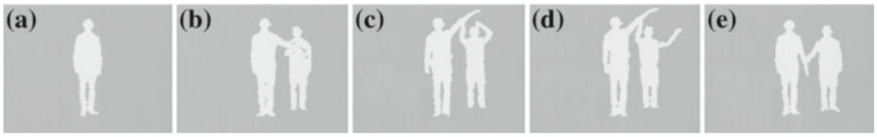


Fig. 14.15 Upper limb training scenario. **a** A patient is captured by the Kinect sensor and given the control. **b** A therapist walks in and helps the patient lift his upper limb. The therapist takes the control of the training game. **c** The therapist waves one of his hands to the *left*. **d** The therapist waves one of his hands to the *right*. **e** The therapist releases the control and the patient finishes the motion

In XIRSH, the gesture control is widely used as an efficient and attracting interaction to motivate the patients. We illustrated two application-level scenarios where the hand gesture recognition results can be directly utilized.

In the first scenario (Fig. 14.15), the patient is asked to lift his upper limb from the body side to conduct the abduction for his shoulder joint. However, this motion might be difficult for an individual patient who just passes level 4 (Brunstrom). Therefore, a physical therapist has to stand by the patient and help the patient's lifting. Two skeletons will be tracked simultaneously through the depth view by the Kinect sensor. The corresponding tracking ID for the patient is not fixed during this process, which causes the incontinence at the application level. As the patient may have the difficulties in motion, a wave hand gesture from the therapist is proposed. Once the waving is detected, the system can identify the situation that there is a therapist who is helping the patient. As a result, the control and focus of the application will be released from the therapist back to the patient. All the corresponding data flows will be correctly led to the patient himself.

In the second scenario (Fig. 14.16), a training is specially designed for exercising the shoulder joint. The patient is immersed into a virtual reality and asked to move the items, such as a clock and a football, into to the correct categories, such as sporting items and living items. During this training, the items are generated at the upper corners of the screen, and the containers are at the bottom corners. The practice is

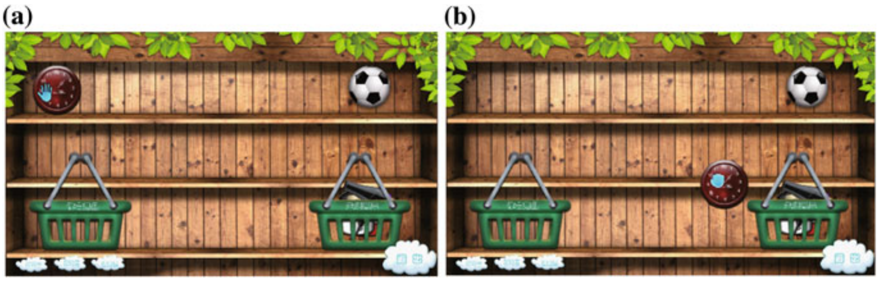


Fig. 14.16 Hand training scenario. **a** A hand over to select the clock (*Basic level*). **b** A grab to pick up the clock and a release to drop it to the living items category (*Advanced level*)

motivating the patient to do some diagonal motions. At the basic level, the patient can move his hand to conduct the mouse cursor movement. A hand over the items for seconds is considered as to select the item. A hand over the corresponding container is to drop the item. The basic training aims to improve the stability and balance. However, some patients with better conditions could be asked to conduct advanced trainings. For example, the hand over gesture is replaced with the hand grab for the picking and the hand release for the dropping. One advantage to motivate the patients to do more practices with fingers and hands.

In both aforementioned scenarios, hand gestures are used to interact with the software programs and provide the biometric feedback such as text prompts or audio reminders. The feedback is very important because it can inform the patients, whether their motions are correct or not. Our experimental results show that the technique proposed the previous section can efficiently recognize the different gestures and send the feedback at a real-time level.

14.9 Summary

We have introduced a novel framework for recognizing hand gestures, which is inspired by the current depth image-based body pose estimation technology. A semi-automatic labeling strategy using a Kinect sensor is described. This procedure is used to generate the samples used for both hand detection, and hand pose estimation. The 3D hand contour with labeled information provides a simplified hand model to facilitate building real-time bare-hand-controlled interface. Our framework can be easily extended to incorporate different desktop applications. We also notice that 3D gesture interaction is not user-friendly enough, because there are still many visual and touch feedbacks needed to improve the realism.

The current work still has several limitations that we plan to focus on in the future work:

(1) Segmenting hand from a long arm or a body is not well handled with the proposed hand extraction and hand parts classification method. Therefore, the form of camera setup is limited.
(2) The accuracy of the contour model is limited by the classification result of hand parts.
(3) To some applications, the classifier needs to be re-trained due to the different application configurations.

Another future research directions include designing new hand partition patterns to improve the recognition accuracy. The contour matching algorithm can be further revised and evaluated to improve the matching accuracy. Moreover, we also would like to develop new hand gesture-controlled applications with the support of this framework and implement a GPU version of the algorithm.

References

1. Riener A (2012) Gestural interaction in vehicular applications. Computer 45(4):42–47
2. Roccetti M, Marfia G, Semeraro A (2012) Playing into the wild: a gesture-based interface for gaming in public spaces. J Vis Commun Image Representat 23(3):426–440
3. Fernandez-Pacheco DG, Albert F, Aleixos N, Conesa J (2012) A new paradigm based on agents applied to free-hand sketch recognition. Exp Syst Appl 39(8):7181–7195
4. Luo C, Chen Y, Krishnan M, Paulik M (2012) The magic glove: a gesture-based remote controller for intelligent mobile robots. In: SPIE on intelligent robots and computer vision: algorithms and techniques, vol 8301, CA, USA, San Francisco
5. Erol A, Bebis G, Nicolescu M, Boyle RD, Twombly X (2007) Vision-based hand pose estimation: a review. CVIU 108(1–2):52–73
6. Zhu Y, Fujimura K (2007) Constrained optimization for human pose estimation from depth sequences. In: Proceedings of the 8th Asian conference on computer vision—volume Part I, November 2007, pp 408–418
7. Siddiqui M, Medioni G (2010) Human pose estimation from a single view point, real-time range sensor. In: IEEE computer society conference on computer vision and pattern recognition workshops (CVPRW), June 2010, pp. 1–8
8. Shotton J, Sharp T, Kipman A, Fitzgibbon A, Finocchio M, Blake A, Cook M, Moore R (2013) Real-time human pose recognition in parts from single depth images. Commun. ACM 56(1):116–124
9. Fanelli G, Gall J, Van Gool L (2011) Real time head pose estimation with random regression forests. In: Proceedings of computer vision and pattern recognition (CVPR), June 2011, pp. 617–624
10. Yao Y, Yao Y, Fu Y (2102) Real-time hand pose estimation from rgb-d sensor. In: IEEE international conference on multimedia and expo (ICME), July 2012, pp 705–710
11. Oikonomidis I, Kyriazis N, Argyros A (2011) Efficient model-based 3d tracking of hand articulations using Kinect. In: BMVC 2011, August. 2011, pp 101.1–101.11
12. de Campos TE, Murray DW (2006) Regression-based hand pose estimation from multiple cameras. In: IEEE computer society conference on computer vision and pattern recognition, June 2006, pp 782–789
13. Xiaohui S, Gang H, Williams L, Ying W (2012) Dynamic hand gesture recognition: an exemplar-based approach from motion divergence fields. Image Vis Comput 30(3):227–235
14. Jun P, Yeo-Lip Y (2006) Led-glove based interactions in multi-modal displays for teleconferencing. In: Proceedings of international conference on artificial reality and telexistence, December 2006, pp 395–399
15. Wang RY, Popović J (2009) Real-time hand-tracking with a color glove. ACM Trans Graph 28:63:1–63:8
16. Sturman DJ, Zeltzer D (1994) A survey of glove-based input. IEEE Comput Graph Appl 14:30–39
17. Mo Z, Lewis JP Neumann U (2005) Smartcanvas: a gesture-driven intelligent drawing desk system. In: Proceedings of international conference on intelligent user interfaces, January 2005, pp 239–243
18. Ogihara A, Matsumoto H, Shiozaki A (2007) Hand region extraction by background subtraction with renewable background for hand gesture recognition. In: international symposium on intelligent signal processing and communications. Japan, Nov, Yonago, pp 227–230
19. Bilal S, Akmeliawati R, El Salami MJ, Shafie AA, and Bouhabba EM (2010) A hybrid method using Haar-like and skin-color algorithm for hand posture detection, recognition and tracking. In: IEEE international conference on mechatronics and automation, August 2010, pp 934–939
20. Chai X, Fang Y, Wang K (2009) Robust hand gesture analysis and application in gallery browsing. In: Proceedings of IEEE ICME, June 2009, pp 938–941
21. Van den Bergh M, Van Gool L (2011) Combining RGB and tof cameras for real-time 3D hand gesture interaction. In: Proceedings IEEE WACV, January 2011, pp 66–72

22. Li H, Greenspan M (2011) Model-based segmentation and recognition of dynamic gestures in continuous video streams. Pattern Recogn 44(8):1614–1628
23. Alon J, Athitsos V, Yuan Q, Sclaroff S (2009) A unified framework for gesture recognition and spatiotemporal gesture segmentation. IEEE Trans PAMI 31(9):1685–1699
24. Guo JM, Liu Y-F, Chang C-H, Nguyen H-S (2012) Improved hand tracking system. IEEE Trans Circ Syst Video Technol. 22(5):693–701
25. Patlolla C, Mahotra S, Kehtarnavaz N (2012) Real-time hand-pair gesture recognition using a stereo webcam. Proceedings IEEE international conference on emerging signal processing applications, pp 135–138
26. Tang M (2011) Recognizing hand gestures with Microsoft's kinect. Tech. Rep. 2011
27. Doliotis P, Stefan A, McMurrough C, Eckhard D, Athitsos V (2011) Comparing gesture recognition accuracy using color and depth information. In: Proceedings of the 4th international conference on PErvasive technologies related to assistive environments, vol 20 May 2011, pp 1–7
28. Minnen D, Zafrulla Z (2011) Towards robust cross-user hand tracking and shape recognition. In: 2011 IEEE international conference on computer vision workshops (ICCV Workshops), November 2011, pp 1235–1241
29. Doliotis P, Athitsos V, Kosmopoulos DI, Perantonis SJ, (2012) Hand shape and 3D pose estimation using depth data from a single cluttered frame. In: Proceedings of international symposium on visual computing (ISVC), vol 7431. April 2012, pp 148–158
30. Oikonomidis I, Kyriazis N, Argyros AA (2012) Tracking the articulated motion of two strongly interacting hands. In: Proceedings of the 2012 IEEE conference on computer vision and pattern recognition (CVPR), June 2012, pp 1862–1869
31. Han J, Shao L, Xu D, Shotton J (2013) Enhanced computer vision with microsoft kinect sensor: a review. IEEE Trans Cybern (T-Cyb), 43(5):1318–1334
32. de La Gorce M, Fleet DJ, Paragios N (2011) Model-based 3d hand pose estimation from monocular video. IEEE Trans PAMI 33:1793–1805
33. Mo Z, Neumann U (2006) Real-time hand pose recognition using low-resolution depth images. In: Proceedings of IEEE CVPR, June 2006, pp 1499–1505
34. Suryanarayan P, Subramanian A, Mandalapu D (2010) Dynamic hand pose recognition using depth data. In: Proceedings IAPR ICPR, December 2010, pp 3105–3108
35. Ren Z, Yuan J, Meng J, Zhang Z (2013) Robust part-based hand gesture recognition using Kinect sensor. IEEE Trans Multimedia 15(5):1110–1120
36. Liu L, Shao L (2013) Learning discriminative representations from RGB-D video data. In: Proceedings of the twenty-third international joint conference on artificial intelligence (IJCAI'13) August 2013, pp 1493–1500
37. Keskin C, Kirac F, Kara YE, Akarun L, (2011) Real time hand pose estimation using depth sensors. In: IEEE workshop on consumer depth cameras for computer vision, November 2011, pp 1228–1234
38. Liang H, Yuan J, Thalmann D, Zhang Z (2013) Model-based hand pose estimation via spatial-temporal hand parsing and 3d fingertip localization. Vis Comput 29(6–8):837–848
39. Stoyanov T, Louloudi A, Andreasson H, Lilienthal AJ (2011) Comparative evaluation of range sensor accuracy in indoor environments. In: Proceedings of the European conference on mobile robots (ECMR), September 2011, pp 19–24
40. Yao Y, and Fu Y (2014) Contour model based hand-gesture recognition using Kinect sensor. IEEE Trans Circ Syst Video Technol, pp 1–1
41. Kramer J, Burrus N, Echtler F, Daniel HC, Parker M (2012) Object modeling and detection. In: Hacking the Kinect, Apress, Berkely
42. Canessa A, Chessa M, Gibaldi A, Sabatini SP, Solari F (2013) Calibrated depth and color cameras for accurate 3d interaction in a stereoscopic augmented reality environment. J Vis Commun Image Represent, 2013, article in Press
43. Herrera DC, Kannala J, Heikkila J (2012) Joint depth and color camera calibration with distortion correction. IEEE Trans Pattern Anal Machine Intell 34(10):2058–2064

44. Wang R, Paris S, Popović J, (2011) Practical color-based motion capture. In: Proceedings of the 2011 ACM SIGGRAPH/Eurographics symposium on computer animation, August 2011, pp 139–146
45. Chai D, Ngan KN (1998) Locating facial region of a head-and-shoulders color image. In: Proceedings of the 3rd international conference on face and gesture recognition, April 1998, pp 124–129
46. Nealen A, Igarashi T, Sorkine O, Alexa M (2006) Laplacian mesh optimization. In: Proceedings of the 4th international conference on computer graphics and interactive techniques in Australasia and Southeast Asia, November 2006, pp 381–389
47. Smiths TF, Waterman MS (1981) Identification of common molecular subsequences. J Mol Biol 147(1):195–197
48. Fjeld M, Fredriksson J, Ejdestig M, Duca F, Boschi K, Voegtli B, Juchli P (2007) Tangible user interface for chemistry education. In: Conference on human factors in computing systems, April 2007, pp 805–808