Contour Model-Based Hand-Gesture Recognition Using the Kinect Sensor

Yuan Yao, *Member, IEEE*, and Yun Fu, *Senior Member, IEEE*

***Abstract*— In RGB-D sensor-based pose estimation, training data collection is often a challenging task. In this paper, we propose a new hand motion capture procedure for establishing the real gesture data set. A 14-patch hand partition scheme is designed for color-based semiautomatic 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 under unconstrained conditions. 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 3-D 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.**

***Index Terms*— Hand gesture recognition, human–computer interaction (HCI), RGB-D sensor.**

I. 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, phones, and personal computers. This trend is not only occurring on the 2-D screen, but also happens in the 3-D world.

However, color images cannot provide enough information for tracking hands in 3-D space because much of the spatial position information has to be inferred and this leads to multiple 2-D–3-D mappings.

New sensors, such as Kinect, Xtion, and Leap Motion, can provide the ability to monitor 3-D motions, and thus make it simple to build systems for HCI via 3-D 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 used 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 data set because the recognition process is sensitive to appearance variations of the target shape and backgrounds. Such data set 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. This makes 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. 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. First, we segment a hand into different parts

Manuscript received July 15, 2013; revised September 29, 2013 and November 4, 2013; accepted January 17, 2014. Date of publication January 28, 2014; date of current version October 29, 2014. This work was supported in part by the National Science Foundation CNS under Award 1314484, in part by the Office of Naval Research under Award N00014-12-1-1028, and in part by the U.S. Army Research Office under Grant W911NF-13-1-0160. This paper was recommended by Associate Editor C. Zhu.

Y. Yao is with the School of Mechatronic Engineering and Automation, Shanghai University, Shanghai 200444, China (e-mail: yaoyuan@shu.edu.cn).

Y. Fu is with the Department of Electrical and Computer Engineering, College of Engineering, and College of Computer and Information Science, Northeastern University, Boston, MA 02115 USA (e-mail: yunfu@ece.neu.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSVT.2014.2302538

and use a 3-D 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 different parts of a hand. To reduce the workload of establishing real training data, we develop a semiautomatic labeling procedure, which uses both RGB data and depth data to label colored hand patches. Finally, we generate the 3-D contour model from the classified pixels. Instead of matching between images, the 3-D 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.

This paper is based on [10] with substantial extensions. A more comprehensive survey and a redesigned training data labeling scheme are presented, as well as a new application framework and an application demonstration. This paper is organized as follows. In Section II, we discuss the related work, especially the depth sensors-based hand gesture recognition. Our hand gesture recognition framework is introduced in Section III. Section IV shows the details of training database collection. Section V describes the process of segmenting hand patches in images. The hand contour model and gesture matching strategy are described in Section VI. Experimental evaluations and the application demonstration are given in Sections VII and VIII, respectively. We finally conclude with future work in Section IX.

II. Related Work

Over the past decades, many hand–gesture-based interaction prototyping systems have been developed. Ali *et al.* [5] reviewed some of this paper. 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, classification-based methods and exemplar-based methods [12].

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 [13], gloves with colored patterns [14], and data glove [15] 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.

However, 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 control 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 consist of hand detection and pose estimation. Therefore, we organize them into two main categories: hand tracking and hand pose estimation, and mainly concern about the methods that use RGB-D sensors.

A. Hand Localization and Tracking

Segmenting a hand from a cluttered background and tracking it steadily and robustly are challenging tasks. The skin color [16] and background subtraction [17] 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 [18], histogram [19], [20], multiscale model [21], and motion cues [22] are employed to overcome this problem. Guo *et al.* [23] presented a method, which 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.

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 [24], scale [19], pixel probability [25], feature probability [10] and the distance to the camera. Paul *et al.* [26] 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 [27] 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.* [28] 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 [29]. Han *et al.* [30] also gave a detailed description of this topic.

B. Hand Pose Estimation

In hand pose estimation, there are a number of methods developed to find a relationship between 2-D hand shape and hand posture [31]. 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 interfaces.

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 indexing technique [22] is employed, which makes it possible to recover hand tracking from each frame. Wang *et al.* [14] provided a real-time hand pose estimation system based on this technique.

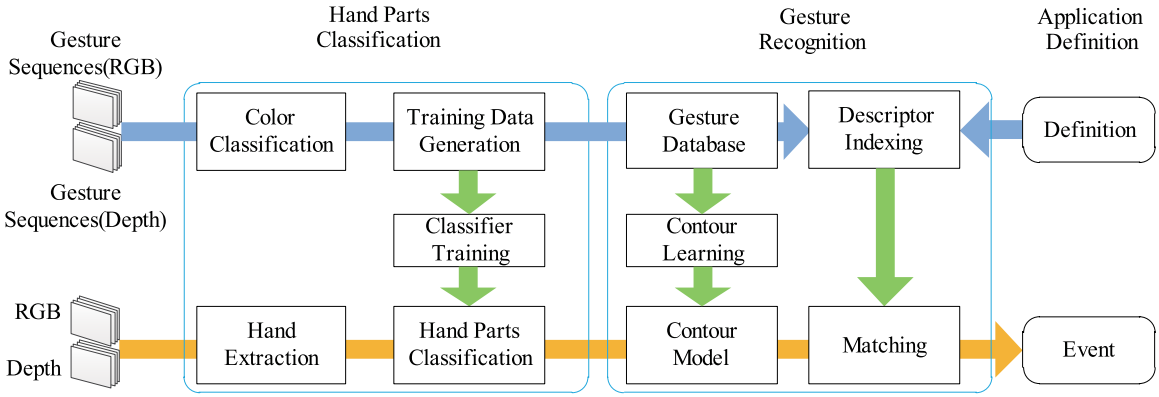


Fig. 1. Framework of our hand gesture recognition system.

To remove the ambiguity generated in the 2-D projection of hand shapes, depth sensors are used. Mo *et al.* [32] used 3-D contours to identify fingers based on a low-resolution depth image. Suryanarayan *et al.* [33] constructed a volume descriptor in depth space and used it to recognize six gestures. Ren *et al.* [34] used a finger shape-based distance to distinguish different hand gestures.

To reconstruct a full degree of freedom hand model, the different parts of a hand must be prelabeled 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.* [35] 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.* [36] 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.

III. Framework Overview

As shown in Fig. 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 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 and 2) the accuracy of Kinect is very close to a laser-based devices within a short range [37]. This makes Kinect a good choice for building gesture driven desktop applications, especially for augmented reality (AR) applications. In addition, Kinect makes it easier to create a labeled training data set using a semiautomatic 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.

A. 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 semiautomatic 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 data set. 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.

B. Gesture Recognition

In the second stage, we use an improved 3-D 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 3-D hand contour model from it in the lower work flow. This model contains not only gesture descriptors but also the 3-D position of the hand. In the demonstration application, we show how they are used for recognizing dynamic gestures.

(a) (b)

Fig. 2. Color glove design. (a) 14 and (b) nine patches.

C. Application Definition

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.

IV. Training Data Generation

Training data collection is always a nontrivial task. Inspired by the current research of color-based motion tracking technique [14], we design two configurations of the color glove. As shown in Fig. 2, the difference is that the one with nine patches cannot identify complex finger motion, therefore, for this paper, we adopt the one with 14 patches.

To collect the training data in a semiautomatic 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 [38]–[40]. 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 multicolor glove and recognize different color regions. We manually initialize the hand location in the color image, 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 [41], which will result in very noisy signals when depth pixel labeling is performed. In our work, the color patches are classified using a Gaussian color model trained in the HSV color space. We then use a predefined graph to validate all pixels, remove the invalidated pixels, and manually finish the final labeling work.

A. 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. Fig. 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. 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.

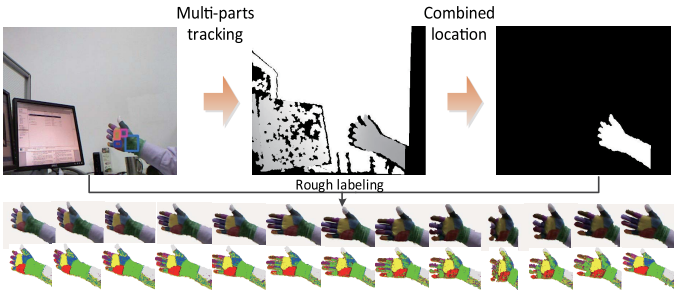


Fig. 3. Hand location and rough labeling.

Simultaneously, 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.

B. Color Classification

Coarsely labeled results need to be further refined. We use a color classification approach [41]. Colors are defined as different 1-D Gaussian distributions $N(\mu_i, \sigma_i)$ in HSV space. We only consider hue, which is estimated from manually labeled samples, usually the first image of each gesture sequence. Every depth pixel of the hand is classified by the following K-means method:

$$C(x) = \arg\min_i \|h_i - \mu_i\|, \quad x \in \text{hand} \tag{1}$$

where h_i is the hue value of the ith depth pixel.

The output of the color classification step is used as training data for generating 3-D hand contours for the gesture database. During the semiautomatic labeling process, there are many incorrectly labeled pixels. After color classification, manually labeling process is still employed to remove this noise. There are 2564 labeled frames of hand samples from 20 people in our training database. Five different types of hand gesture sequences are included in the database. Using the classification results, the average labeling time is decreased from 6 min to 30 s for each image in our experiments.

V. 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 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 processes 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.

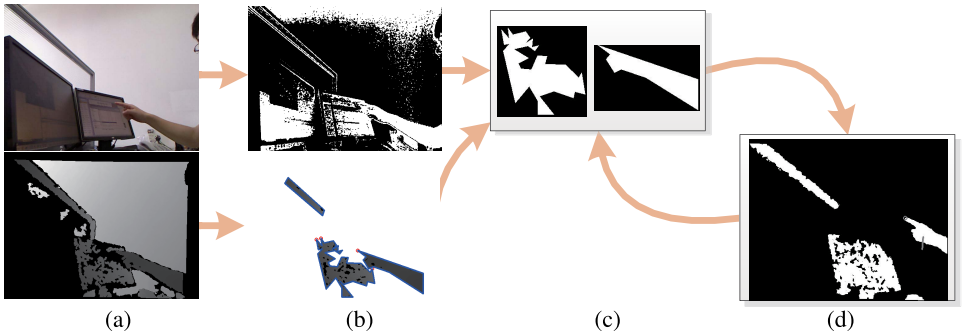


Fig. 4. Hands extraction. (a) RGB and depth images. (b) Initial segmentation. (c) Targets classification. (d) Hand tracking.

A. Hand Extraction

To discriminate hand-like objects from other foreground objects, a tracking technique is used. The complete procedure is shown in Fig. 4. For RGB images, the skin color pixels are extracted from RGB image by [42] to generate the skin color mask [as shown in Fig. 4(b) upper].

For the depth image, we first assume that hands move within a certain distance range to the camera (distance <1.6 meters). Outside this range, part of the depth data are 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. 4(a) bottom left]. 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 2-D 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 [16] to track the trajectory of the palm center in 2-D 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.

We use a 2-D 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

$$\operatorname{acos}\left(\left(x_{i-1}^{A}-x_{i}^{A,H}\right)\cdot\left(x_{i+1}^{A}-x_{i}^{A,H}\right)\right)<T \tag{2}$$

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. 5(a), 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. 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.

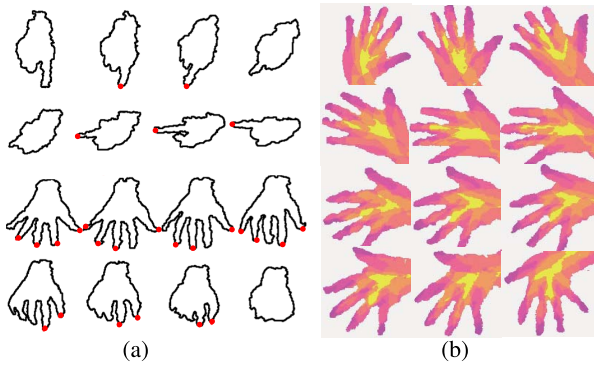


Fig. 5. Features. (a) 2-D shape features defined on hand contour. The red points indicate the locations of features. (b) Position feature in a rotated hand.

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

$$P(h=1|\mathcal{S})=\frac{P(\mathcal{S}|h=1)P(h=1)}{P(\mathcal{S})} \tag{3}$$

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 $P(\mathcal{S}|h = 1)$ and $P(\mathcal{S}|h = 0)$ from the training database. Here, $P(\mathcal{S}) = P(\mathcal{S}|h = 1)P(h = 1) + P(\mathcal{S}|h = 0)P(h = 0)$. Candidate regions are chosen if their posterior probability is greater than a threshold.

B. 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.

To overcome this problem, we create a new feature, which is defined over the 3-D 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

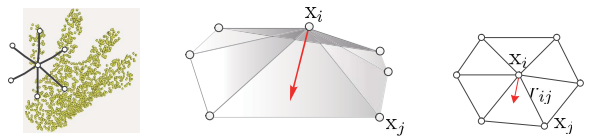


Fig. 6. Position feature is a geometrical feature defined on the depth image (left), which represents local details. The red arrow's direction approximates the normal; the size approximates the mean curvature (middle). In the 2-D depth image, this feature can be computed by a neighborhood sampling descriptor (right).

belongs to. As shown in Fig. 6(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 $X = [x_1, x_2, \ldots, x_n]$, $x_i = \{x_{iu}, y_{iv}, d_I(x_i)\}$ is the depth pixels set. $d_I(x_i)$ is the depth value of pixel x_i. Inspired by [43], we define the position feature for each pixel x_i by

$$f_i(I, x) = \sum_{i,j \in \mathrm{E}} \omega_{i,j} \left(x_j - x_i\right) \tag{4}$$

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

A graphical description of this position feature is shown by Fig. 6 (middle). The neighbor domain information is introduced by the topology. In (4), the direction of $f_i(I, x)$ approximates the pixel's normal, and the size of $f_i(I, x)$ represents the mean curvature. We only use the size, thus the position feature is rotation invariant. 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 x_i and x_j (Fig. 6 right). Using a constant $r_{i,j}$, we can define the position feature on the depth image as

$$\delta_i = L\left(x_i\right) = d_I(x_i) - \frac{1}{n} \sum_{j \in E} s_I(x_j) \tag{5}$$

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

$$s_I(x) = \begin{cases} d_I(x), & x \in \text{hand} \\ b, & x \notin \text{hand}. \end{cases} \tag{6}$$

Fig. 5(b) shows 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 observed in Fig. 5(b), 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 2-D 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. Fig. 7(b) 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

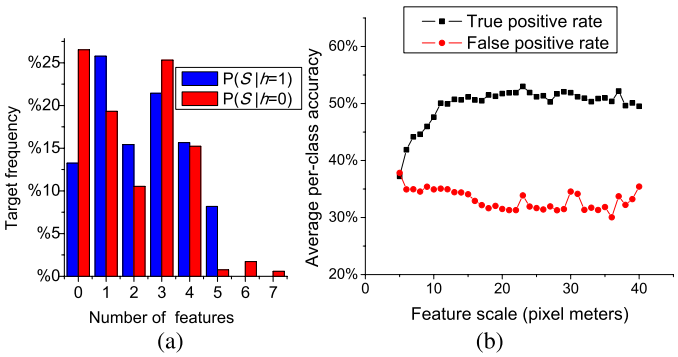


Fig. 7. (a) Prior distribution of shape features. (b) Feature scale and average per-classes accuracy.

the false positive rate, we refine the estimation results with a contour model in the gesture recognition stage.

VI. Gesture Recognition

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

A. Contour Model

The final output of the estimation process is a labeled 3-D contour. There are several advantages of using labeled 3-D 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 3-D skin model.

1) Contour Descriptor: The hand contour model C is represented by a ordered list of points on the contour $\mathbf{c} = \{\mathbf{v}_1^c, \mathbf{v}_2^c, \ldots, \mathbf{v}_n^c\}$ with its corresponding label vector $\mathbf{l} = \{l_1, l_2, \ldots, l_n\}$. 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 3-D coordinate sequence and labeling index sequence cannot provide a rotation and scale invariance. We add another sequence, $\mathbf{m} = \{m_1, m_2, \ldots, m_n\}$, for the contour representation. $\mathbf{m}$ is a normalized length sequence

$$m_i = \frac{\text{Length}(\{\mathbf{v}_j^c | l_j\})}{\text{Length}(\mathbf{c})} \tag{7}$$

where j represents the continuous indexing. The descriptor of the ith hand contour C^i is represented by the

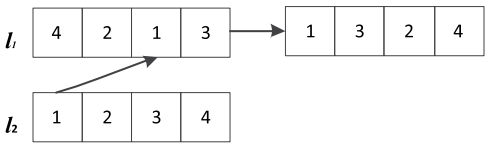


Fig. 8. Contour descriptor alignment.

$\{l_1^i m_1^i, l_2^i m_2^i, \ldots, l_d^i m_d^i\}$, where d is the dimension of the descriptor. By selecting d most significant m_i, d can be set to a fixed dimension.

B. Contour Generation

Given a 3-D 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

$$P(\mathbf{c}|\mathbf{v}_i^c) = \frac{1}{n}\gamma_i \sum_{j\in N(i)} P_j(\mathbf{c}|\mathbf{v}_j) \tag{8}$$

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 $P(\mathbf{c}|x)$. γ_i is a learning parameter defined as

$$\gamma_i = \frac{p_c^i}{M} \sum_{k\in\varsigma} P(\mathbf{c}|\mathbf{v}_k^c) \tag{9}$$

which considers the continuity of the 3-D 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. Using this model, the gesture recognition process can be described as a sequence matching problem, which can also be used to recover the 3-D hand position.

C. Contour Matching

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. 8. That means given a contour descriptor C^i, the query result from a database should satisfy

$$C^i = \underset{i\in\Psi, j\in\Omega}{\arg\min}\, D(C^i, C^j)w \tag{10}$$

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

$$w = \sum_{k=\{1,\ldots,d\}} l_k^i \;\texttt{xor}\; l_k^j. \tag{11}$$

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 [44] 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.

VII. Validation

We evaluated our method on the data set collected in Section IV. 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 data set provided in [34].

A. Hand Detection

In most of the indoor bare–hand interaction applications, hands and objects often appear together. Therefore, we assume $P(h=1) = P(h=0) = 0.5$. Thus (3) can be simplified to

$$P(h=1|\mathcal{S}) = \frac{P(\mathcal{S}|h=1)}{P(\mathcal{S}|h=1) + P(\mathcal{S}|h=0))}. \tag{12}$$

The distribution of $P(\mathcal{S}|h=1)$ is learned from 814 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 $P(\mathcal{S}|h=0)$. These two distributions are shown in Fig. 7(a), 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 predefined threshold. The number of shape features is counted and used to compute (3). This method is tested on the remaining 1750 images containing a cluttered background. The hand detection procedure achieves 56.8% accuracy. After using the skin color segmentation described in Section V, the accuracy increases to 95.43%.

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; and 3) a person is wearing gloves. Body skeletal trackers can be used to solve these problems.

B. Hand Parts Classification

In hand parts classification experiment, we select 434 frames from two subjects for training, and the frames from the other 18 subjects for testing. The training samples and testing samples are all collected from the same laboratory environment, but with different backgrounds and camera view angles.

Fig. 7(b) shows the relationship between the 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 Section V as the position feature. Compared with the per-part recognition accuracy rate in [10], the classification results show a significant improvement. This curve will change slightly when different topologies are selected for shape feature.

TABLE I
PER-PIXEL HAND PARTS CLASSIFICATION RESULTS

Category	sample quantity	true positive	false positive
Thumb T	575612	31.1%	0.66%
Thumb	1048300	40.0%	35.30%
Index T	613732	45.0%	8.70%
Index	819580	24.9%	1.40%
Middle T	781460	68.5%	40.20%
Middle	884384	69.6%	72.60%
Ring T	689972	60.0%	61.30%
Ring	804332	58.3%	54.00%
Small T	590860	31.6%	7.42%
Small	529868	24.8%	12.90%
CMC	2919992	64.0%	46.70%
Palm	2740828	84.6%	84.90%
TM	1696340	27.2%	3.93%
ARM	10448692	96.4%	8.26%
Average accuracy		51.87%	31.31%

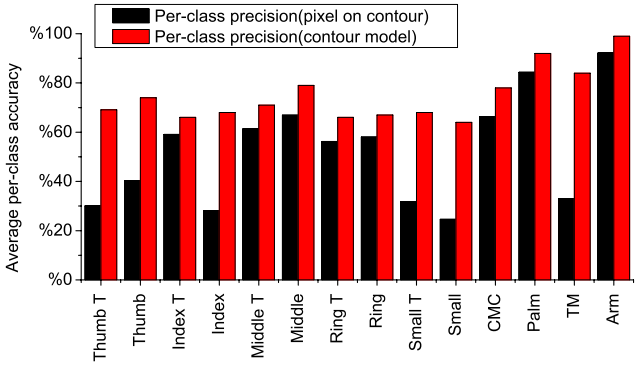


Fig. 9. Per-pixel and per-segment classification accuracies with contour model.

Table I shows the per-pixel classification results of detecting different hand patches. The experiment is done on 1906 frames, where T indicates finger tip part, CMC indicates the region near the root of little finger, and TM indicates the region near the root of thumb. The average accuracy is improved to 51.87%. 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 overfitting problem.

Fig. 9 shows the per-class accuracy for pixels located on the hand contour (Fig. 9 black) and the per-class accuracy by using the contour model (Fig. 9 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 52.31% to 74.65%. The error rate is still large but hand orientation can be estimated from the contour, which can also be used in gesture matching.

C. Gestures Recognition

Our method is tested on the database provided in [34] 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 data set of [34] is used for testing. Fig. 10 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 using 40 training samples. Although the recognition rate of some categories has not been significantly improved, the average true positive rate is better.

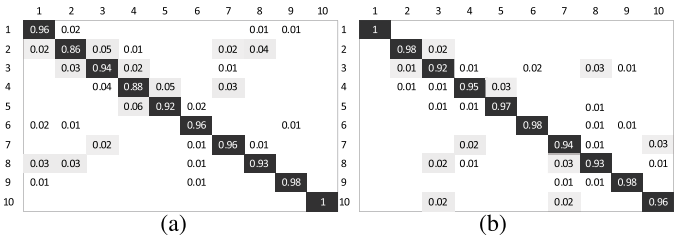


Fig. 10. Confusion matrix of gestures recognition results. The gesture categories are ordered by the original database. (a) Near-convex decomposition + FEMD [34]. (b) Hand contour matching.

We use a desktop computer with an AMD Phenom II X4, 3.0-GHz CPU with 3-G 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 35 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.

VIII. SYSTEM APPLICATION

To evaluate the feasibility and effectiveness of the proposed framework, we perform case studies with a HCI program named AR Chemical [45], which is an application that uses a tangible user interface for organic chemistry education. The interface uses an AR mouse and AR makers to allow users to compose and directly interact with 3-D molecular models. All the interactions are done on a real desk. The operation includes moving, dragging, positioning and assembling virtual structures in a 3-D space. Therefore, delicate and complex interactive behaviors and hand position information are needed.

We simulate this system and provide four hand gestures to replace the original operation that used with the AR paddle and computer mouse. First, we use hand grab gesture to replace the AR paddle 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 3-D 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 setup two gesture buffers in the application.

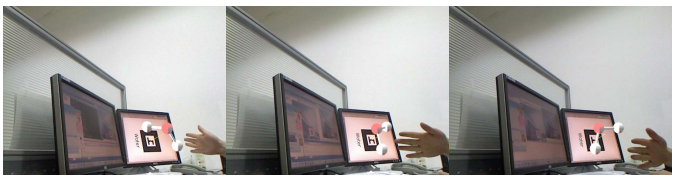


Fig. 11. Waving gesture for rotation control.

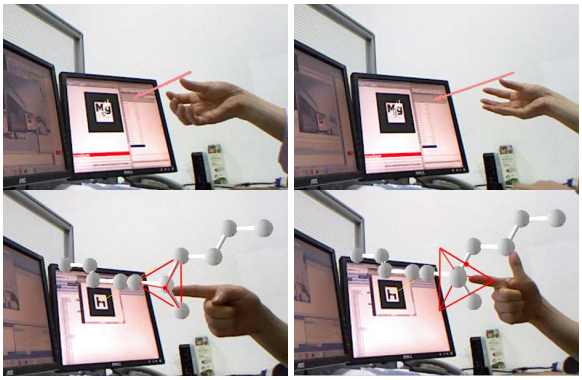


Fig. 12. Sample frames from the atom grabbing and molecular assembling operation.

One for reserving sequential contour with 3-D 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. 11 shows the rotation operation. Fig. 12 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.

IX. Conclusion

We have introduced a novel framework for recognizing hand gestures, which is inspired by the current depth image-based body pose estimation technology. A semiautomatic 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 3-D 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 3-D gesture interaction is not user-friendly enough, because there are still many visual and touch feedbacks needed to improve the realism.

This paper 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.

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.

References

[1] A. Riener, "Gestural interaction in vehicular applications," *Computer*, vol. 45, no. 4, pp. 42–47, Apr. 2012.

[2] M. Roccetti, G. Marfia, and A. Semeraro, "Playing into the wild: A gesture-based interface for gaming in public spaces," *J. Vis. Commun. Image Represent.*, vol. 23, no. 3, pp. 426–440, Apr. 2012.

[3] D. G. Fernandez-Pacheco, F. Albert, N. Aleixos, and J. Conesa, "A new paradigm based on agents applied to free-hand sketch recognition," *Expert Syst. Appl.*, vol. 39, no. 8, pp. 7181–7195, Jun. 2012.

[4] C. Luo, Y. Chen, M. Krishnan, and M. Paulik, "The magic glove: A gesture-based remote controller for intelligent mobile robots," *Proc. SPIE*, vol. 8301, pp. 1–11, Dec. 2012.

[5] A. Erol, G. Bebis, M. Nicolescu, R. D. Boyle, and X. Twombly, "Vision-based hand pose estimation: A review," *Comput. Vis. Image Understand.*, vol. 108, nos. 1–2, pp. 52–73, Nov. 2007.

[6] Y. Zhu and K. Fujimura, "Constrained optimization for human pose estimation from depth sequences," in *Proc. 8th Asian Conf. Comput. Vis.*, vol. 1. Nov. 2007, pp. 408–418.

[7] M. Siddiqui and G. Medioni, "Human pose estimation from a single view point, real-time range sensor," in *Proc. IEEE Comput. Soc. Conf. CVPRW*, Jun. 2010, pp. 1–8.

[8] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, *et al.*, "Real-time human pose recognition in parts from single depth images," *Commun. ACM*, vol. 56, no. 1, pp. 116–124, Jan. 2013.

[9] G. Fanelli, J. Gall, and L. Van Gool, "Real time head pose estimation with random regression forests," in *Proc. CVPR*, Jun. 2011, pp. 617–624.

[10] Y. Yao, Y. Yao, and Y. Fu, "Real-time hand pose estimation from RGB-D sensor," in *Proc. IEEE ICME*, Jul. 2012, pp. 705–710.

[11] I. Oikonomidis, N. Kyriazis, and A. Argyros, "Efficient model-based 3D tracking of hand articulations using Kinect," in *Proc. BMVC*, Aug. 2011, pp. 1–11.

[12] S. Xiaohui, H. Gang, L. Williams, and W. Ying, "Dynamic hand gesture recognition: An exemplar-based approach from motion divergence fields," *Image Vis. Comput.*, vol. 30, no. 3, pp. 227–235, 2012.

[13] P. Jun and Y. Yeo-Lip, "Led-glove based interactions in multi-modal displays for teleconferencing," in *Proc. Int. Conf. Artif. Real. Telexistence*, Dec. 2006, pp. 395–399.

[14] R. Y. Wang and J. Popović, "Real-time hand-tracking with a color glove," *ACM Trans. Graph.*, vol. 28, pp. 1–8, Jul. 2009.

[15] D. J. Sturman and D. Zeltzer, "A survey of glove-based input," *IEEE Comput. Graph. Appl.*, vol. 14, no. 1, pp. 30–39, Jan. 1994.

[16] Z. Mo, J. P. Lewis, and U. Neumann, "Smartcanvas: A gesture-driven intelligent drawing desk system," in *Proc. Int. Conf. Intell. User Inter.*, Jan. 2005, pp. 239–243.

[17] A. Ogihara, H. Matsumoto, and A. Shiozaki, "Hand region extraction by background subtraction with renewable background for hand gesture recognition," in *Proc. Int. Symp. Intell. Signal Process. Commun.*, Yonago, Japan, Nov. 2007, pp. 227–230.

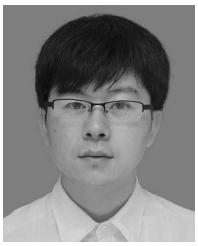
[18] S. Bilal, R. Akmeliawati, M. J. El Salami, A. A. Shafie, and E. M. Bouhabba, "A hybrid method using Haar-like and skin-color algorithm for hand posture detection, recognition and tracking," in *Proc. IEEE Int. Conf. Mechatron. Autom.*, Aug. 2010, pp. 934–939.

[19] X. Chai, Y. Fang, and K. Wang, "Robust hand gesture analysis and application in gallery browsing," in *Proc. IEEE ICME*, Jun. 2009, pp. 938–941.

[20] M. Van den Bergh and L. Van Gool, "Combining RGB and ToF cameras for real-time 3D hand gesture interaction," in *Proc. IEEE WACV*, Jan. 2011, pp. 66–72.

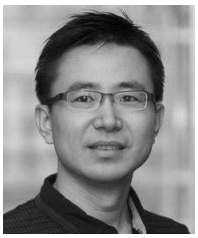
[21] H. Li and M. Greenspan, "Model-based segmentation and recognition of dynamic gestures in continuous video streams," *Pattern Recognit.*, vol. 44, no. 8, pp. 1614–1628, Aug. 2011.

[22] J. Alon, V. Athitsos, Q. Yuan, and S. Sclaroff, "A unified framework for gesture recognition and spatiotemporal gesture segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 9, pp. 1685–1699, Sep. 2009.
[23] J. M. Guo, Y.-F. Liu, C.-H. Chang, and H.-S. Nguyen, "Improved hand tracking system," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 5, pp. 693–701, May 2012.
[24] C. Patlolla, S. Mahotra, and N. Kehtarnavaz, "Real-time hand-pair gesture recognition using a stereo webcam," in *Proc. IEEE Int. Conf. Emerging Signal Process. Appl.*, Jan. 2012, pp. 135–138.
[25] M. Tang, "Recognizing hand gestures with Microsoft's Kinect," Dept. Electr. Eng., Stanford Univ., Stanford, CA, USA, Tech. Rep., 2011.
[26] P. Doliotis, A. Stefan, C. McMurrough, D. Eckhard, and V. Athitsos, "Comparing gesture recognition accuracy using color and depth information," in *Proc. 4th Int. Conf. Pervas. Technol. Rel. Assistive Environ.*, May 2011, pp. 1–7.
[27] D. Minnen and Z. Zafrulla, "Towards robust cross-user hand tracking and shape recognition," in *Proc. IEEE ICCV Workshops*, Nov. 2011, pp. 1235–1241.
[28] P. Doliotis, V. Athitsos, D. I. Kosmopoulos, and S. J. Perantonis, "Hand shape and 3D pose estimation using depth data from a single cluttered frame," in *Proc. ISVC*, Apr. 2012, pp. 148–158.
[29] I. Oikonomidis, N. Kyriazis, and A. A. Argyros, "Tracking the articulated motion of two strongly interacting hands," in *Proc. IEEE CVPR*, Jun. 2012, pp. 1862–1869.
[30] J. Han, L. Shao, D. Xu, and J. Shotton, "Enhanced computer vision with microsoft Kinect sensor: A review," *IEEE Trans. Cybern.*, vol. 43, no. 5, pp. 1318–1334, Oct. 2013.
[31] M. de La Gorce, D. J. Fleet, and N. Paragios, "Model-based 3D hand pose estimation from monocular video," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 9, pp. 1793–1805, Sep. 2011.
[32] Z. Mo and U. Neumann, "Real-time hand pose recognition using low-resolution depth images," in *Proc. IEEE CVPR*, Jun. 2006, pp. 1499–1505.
[33] P. Suryanarayan, A. Subramanian, and D. Mandalapu, "Dynamic hand pose recognition using depth data," in *Proc. IAPR ICPR*, Dec. 2010, pp. 3105–3108.
[34] Z. Ren, J. Yuan, J. Meng, and Z. Zhang, "Robust part-based hand gesture recognition using Kinect sensor," *IEEE Trans. Multimedia*, vol. 15, no. 5, pp. 1110–1120, Aug. 2013.
[35] C. Keskin, F. Kirac, Y. E. Kara, and L. Akarun, "Real time hand pose estimation using depth sensors," in *Proc. IEEE Workshop Consum. Depth Cameras Comput. Vis.*, Nov. 2011, pp. 1228–1234.
[36] H. Liang, J. Yuan, D. Thalmann, and Z. Zhang, "Model-based hand pose estimation via spatial-temporal hand parsing and 3D fingertip localization," *Vis. Comput.*, vol. 29, nos. 6–8, pp. 837–848, Jun. 2013.
[37] T. Stoyanov, A. Louloudi, H. Andreasson, and A. J. Lilienthal, "Comparative evaluation of range sensor accuracy in indoor environments," in *Proc. ECMR*, Sep. 2011, pp. 19–24.
[38] J. Kramer, N. Burrus, F. Echtler, H. C. Daniel, and M. Parker, "Object modeling and detection," in *Hacking the Kinect*. Berkeley, CA, USA: Springer-Verlag, 2012.
[39] A. Canessa, M. Chessa, A. Gibaldi, S. P. Sabatini, and F. Solari, "Calibrated depth and color cameras for accurate 3D interaction in a stereoscopic augmented reality environment," *J. Vis. Commun. Image Represent.*, vol. 25, no. 1, pp. 227–237, Jan. 2014.
[40] C. Daniel Herrera, J. Kannala, and J. Heikkila, "Joint depth and color camera calibration with distortion correction," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 10, pp. 2058–2064, Oct. 2012.
[41] R. Wang, S. Paris, and J. Popović, "Practical color-based motion capture," in *Proc. ACM SIGGRAPH/Eurograph. Symp. Comput. Autom.*, Aug. 2011, pp. 139–146.
[42] D. Chai and K. N. Ngan, "Locating facial region of a head-and-shoulders color image," in *Proc. 3rd Int. Conf. Face Gesture Recognit.*, Apr. 1998, pp. 124–129.
[43] A. Nealen, T. Igarashi, O. Sorkine, and M. Alexa, "Laplacian mesh optimization," in *Proc. 4th Int. Conf. Comput. Graph. Interact. Tech. Austral. Southeast Asia*, Nov. 2006, pp. 381–389.
[44] T. F. Smite and M. S. Waterman, "Identification of common molecular subsequences," *J. Mol. Biol.*, vol. 147, no. 1, pp. 195–197, Mar. 1981.
[45] M. Fjeld, J. Fredriksson, M. Ejdestig, F. Duca, K. Boschi, B. Voegtli, *et al.*, "Tangible user interface for chemistry education," in *Proc. Conf. Human Factors Comput. Syst.*, Apr. 2007, pp. 805–808.



Yuan Yao (S'05–M'08) received the B.Eng. degree in mechanical engineering and the M.Eng. degree in mechanical design from Xi'an Polytechnic University, Xi'an, China, and the Ph.D. degree in computer science from Zhejiang University, Hangzhou, China.

He was a Post-Doctoral with Rapid Manufacture Engineering Center, Shanghai University, Shanghai, China, from 2006 to 2008. He is currently an Associate Professor with the School of Mechatronic Engineering and Automation, Shanghai University. From 2010 to 2011 he was a Visiting Scholar with the Department of Computer Science and Engineering, State University of New York, Buffalo, NY, USA. His research interests include computer vision, computer aided design, and computer aided tissue engineering.



Yun Fu (S'07–M'08–SM'11) received the B.Eng. degree in information engineering and the M.Eng. degree in pattern recognition and intelligence systems from Xi'an Jiaotong University, Xi'an, China, and the M.S. degree in statistics and the Ph.D. degree in electrical and computer engineering from University of Illinois at Urbana-Champaign, Urbana, IL, USA.

He was a Scientist with BBN Technologies, Cambridge, MA, USA, from 2008 to 2010. He was a Part-Time Lecturer with the Department of Computer Science, Tufts University, Medford, MA, in 2009. He was a tenure-track Assistant Professor with the Department of Computer Science and Engineering, State University of New York, Buffalo, NY, USA, from 2010 to 2012. He has been an interdisciplinary tenure-track Assistant Professor with the College of Engineering and the College of Computer and Information Science, Northeastern University, Boston, MA, since 2012. He has extensive publications in leading journals, books/book chapters, and international conferences/workshops. His research interests include interdisciplinary research in machine learning, social media analytics, human–computer interaction, and cyber–physical systems.

Dr. Fu is an Associate Editor, Chair, PC member, and reviewer of many top journals and international conferences/workshops. He received the Rockwell Automation Master of Science Award in 2002, Edison Cups of the GE Fund Edison Cup Technology Innovation Competition in 2002, the Hewlett-Packard Silver Medal and Science Scholarship in 2003, the Chinese Government Award for Outstanding Self-Financed Students Abroad in 2007, the IEEE International Conference on Image Processing Best Paper Award in 2007, the Beckman Graduate Fellowship from 2007 to 2008, the M. E. Van Valkenburg Graduate Research Award in 2008, the ITESOFT Best Paper Award of IAPR International Conferences on the Frontiers of Handwriting Recognition in 2010, the Google Faculty Research Award in 2010, the IEEE International Conference on Multimedia and Expo Quality Reviewer in 2011, the IEEE ICDM Workshop on Large Scale Visual Analytics Best Paper Award in 2011, the IC Post-Doctoral Research Fellowship Award in 2011, the IEEE TCSVT Best Associate Editor Award in 2012, the IEEE FG Best Student Paper Honorable Mention Award in 2013, and the INNS Young Investigator Award in 2014. His research is broadly supported by NSF, DOD, DARPA, IARPA, ONR, AFOSR, ARL/ARO, NGA, IC, and Google. He is as an Associate Editor of IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEANING SYSTEMS and IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY. He is a Lifetime Member of ACM, AAAI, SPIE, and the Institute of Mathematical Statistics, and was a Beckman Graduate Fellow from 2007 to 2008.