Frontiers of Information Technology & Electronic Engineering
www.zju.edu.cn/jzus; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Extracting hand articulations from monocular depth images using curvature scale space descriptors*

Shao-fan WANG†1, Chun LI[1], De-hui KONG†‡1, Bao-cai YIN[2,1,3]

([1]*Beijing Key Laboratory of Multimedia and Intelligent Software Technology, College of Metropolitan Transportation, Beijing University of Technology, Beijing 100124, China*)

([2]*School of Software Technology, Dalian University of Technology, Dalian 116024, China*)

([3]*Collaborative Innovation Center of Electric Vehicles in Beijing, Beijing 100081, China*)

†E-mail: wangshaofan@bjut.edu.cn; kdh@bjut.edu.cn

Received Apr. 20, 2015; Revision accepted Oct. 26, 2015; Crosschecked Dec. 9, 2015

Abstract: We propose a framework of hand articulation detection from a monocular depth image using curvature scale space (CSS) descriptors. We extract the hand contour from an input depth image, and obtain the fingertips and finger-valleys of the contour using the local extrema of a modified CSS map of the contour. Then we recover the undetected fingertips according to the local change of depths of points in the interior of the contour. Compared with traditional appearance-based approaches using either angle detectors or convex hull detectors, the modified CSS descriptor extracts the fingertips and finger-valleys more precisely since it is more robust to noisy or corrupted data; moreover, the local extrema of depths recover the fingertips of bending fingers well while traditional appearance-based approaches hardly work without matching models of hands. Experimental results show that our method captures the hand articulations more precisely compared with three state-of-the-art appearance-based approaches.

Key words: Curvature scale space (CSS), Hand articulation, Convex hull, Hand contour
http://dx.doi.org/10.1631/FITEE.1500126 **CLC number:** TP391; TP751

1 Introduction

Extracting human hand articulations such as fingertips, finger-knuckles, finger-roots, and hand contours is an interesting and important task, which has various applications in human-computer interaction and virtual reality. This task is challenging because human hands, like other articulable objects, have many degrees of freedom, constrained parameter space, and suffer self-occlusion. Although special hardware such as data gloves and the Kinect sensor has been successfully developed, it either needs to be worn or produces a lower precision.

Research on detecting hand articulations can be divided into two categories: appearance-based approaches (Rosales *et al.*, 2001; Athitsos and Sclaroff, 2002; 2003; Tomasi *et al.*, 2003; Schlattmann *et al.*, 2007; Feng *et al.*, 2011; Lee and Lee, 2011; Ren *et al.*, 2011; Cerezo, 2012; Nagarajan *et al.*, 2012; Maisto *et al.*, 2013) and model-based approaches (Chang *et al.*, 2008; de La Gorce *et al.*, 2011; Keskin *et al.*, 2011; Oikonomidis *et al.*, 2011; Kirac *et al.*, 2014; Ma and Wu, 2014; Morshidi and Tjahjadi, 2014; Qian *et al.*, 2014; Tompson *et al.*, 2014). Appearance-based approaches, also known as discriminative approaches,

‡ Corresponding author

* Project supported by the National Natural Science Foundation of China (Nos. 61227004, 61370120, 61390510, 61300065, and 61402024), Beijing Municipal Natural Science Foundation, China (No. 4142010), Beijing Municipal Commission of Education, China (No. km201410005013), and the Funding Project for Academic Human Resources Development in Institutions of Higher Learning under the Jurisdiction of Beijing Municipality, China

ORCID: Shao-fan WANG, http://orcid.org/0000-0002-3045-624X

©Zhejiang University and Springer-Verlag Berlin Heidelberg 2016

learn a mapping from the space of image features to the space of hand configurations and formulate the task as a supervised classification problem. While appearance-based approaches give effective computations, such approaches show a limitation of faithfully capturing hand pose because hand configurations can hardly be completely recovered using the features of images.

Model-based approaches, also known as generative approaches, estimate hand states by matching the kinematic structure with image features and formulate the task as a high-dimensional search problem. While model-based approaches capture hand configurations more precisely, the high dimensionality of solution spaces and high nonlinearity of optimization functions make the computation highly expensive and prevent them from being applied to real-time systems.

Although some model-based approaches (Maisto *et al.*, 2013; Kirac *et al.*, 2014; Ma and Wu, 2014; Qian *et al.*, 2014; Tompson *et al.*, 2014) propose detecting hand articulations in real time using various techniques for solving optimization models or using efficient searching algorithms, complicated model parameters are difficult to obtain in a both precise and efficient fashion. This paper proposes an appearance-based method for hand articulation detection from a monocular depth image. While some appearance-based methods use an angle detector or a convex hull detector of the contour points of hand images for extracting local maxima, our method uses the 2D curvature scale space (CSS) descriptor for the task. The CSS descriptor (Abbasi *et al.*, 1999) is a multiscale description of the invariant local features of 2D shapes, which is standardized in MPEG-7 as one of the most important shape descriptors. Instead of using the CSS descriptor for shape matching, we extract the fingertips and finger-valleys of straight fingers using the CSS descriptor with modified curvature thresholds. The fingertips and finger-valleys of bending fingers which do not appear at the hand contour are then detected using predefined angle or depth thresholds of detected fingers. We compare our method with three state-of-the-art appearance-based methods, and show that our method detects the fingertips more precisely because the CSS descriptor is more robust to noisy or corrupted depth data.

2 Related work

This section reviews previous works on hand feature extraction, including appearance-based approaches and model-based approaches.

2.1 Appearance-based approaches

Rosales *et al.* (2001) proposed a state recovering method by learning the mapping between hand joint configurations and the corresponding visual features which are generated from a computer graphics module. Athitsos and Sclaroff (2002) retrieved the most similar matches hierarchically from a large database of synthetic hand images, and obtained both the ground truth labels of those matches and camera viewpoint information. They further proposed an image-to-model chamfer distance and a probabilistic line matching method, and formulated hand pose estimation as an image database indexing problem (Athitsos and Sclaroff, 2003). Tomasi *et al.* (2003) tracked hand gestures with fast and complex motions, by imposing the Fourier shape descriptor on hand silhouettes and interpolating missing data of each frame. Schlattmann *et al.* (2007) extracted the protruding fingertips from the visual hull of segmented images of different cameras, and estimated hand gestures using the position and orientation of hands. Feng *et al.* (2011) extracted hand features by first estimating shape features using an approximated polygon of the hand contour, and then refining the shape features by expressing images in a multiscale space to obtain the response strength of different features. Similarly, Maisto *et al.* (2013) computed the convex envelope of hand contours and tracked the fingertips using both a center-of-mass-based variation and a geometry-based variation. Lee and Lee (2011) proposed a scale-invariant angle detector to locate fingertips and finally recognized fingertip actions using hand contours. Ren *et al.* (2011) proposed a novel distance metric for hand dissimilarity measure for matching merely fingers instead of the whole hand shape. Nagarajan *et al.* (2012) proposed a hand gesture recognition framework by detecting the hand skin color with the HSV color space and morphological operations and locating fingertips with the convex hull of hand contours. Cerezo (2012) detected the fingertips using the threshold of angles formed by vectors

generated from contour points of the hand, followed by a normalization of the 2D points within the depth image for obtaining 3D locations of fingertips.

2.2 Model-based approaches

Chang *et al.* (2008) proposed an appearance-guided particle filtering method for high degree-of-freedom hand tracking, which formulates the tracking problem as finding the maximum a posteriori solution of a probability propagation model consisting of several state space vectors. de La Gorce *et al.* (2011) proposed an analysis-by-synthesis approach for tracking moving hands by incorporating both shading and texture information while handling self-occlusion. The shading information was captured by using a triangulated mesh-based model and the texture estimation was completed using the same objective function used for tracking with a smoothness regularization term. Oikonomidis *et al.* (2011) proposed a 3D hand model consisting of parametrized geometric primitives represented by two basic primitives: sphere and truncated cylinder, and estimated model parameters by minimizing the discrepancy between predicted features and observed features using particle swarm optimization. Morshidi and Tjahjadi (2014) proposed another method, a gravity optimized particle filter, for solving the model of hand configurations, where the localization and labeling of fingers are extracted using convexity defects. Ma and Wu (2014) proposed a hand tracking system using depth sequences which consists of three phases: hand segmentation, k-nearest neighbor database searching, and an improved particle swarm optimization. The searching is fulfilled using both the location of all bulges in the z-direction and a binary coding of the shape of segmented hand masks as two features. Qian *et al.* (2014) combined gradient-based optimization and stochastic optimization to model human hands in real time, using a number of spheres and a cost function involving the alignment and relative position of the point cloud with respect to the sphere model. Keskin *et al.* (2011), Kirac *et al.* (2014), and Tompson *et al.* (2014) classified pixels of depth images using random decision forests, with a different hierarchy of modes over articulation points of hands.

3 Curvature scale space based hand articulation extraction

3.1 Extracting hand contour and palm center

We illustrate the main procedure for extracting the contour of the hand and the palm center in Fig. 1, which includes six steps: (1) find a hand point on the depth image using the function hand_points of openNI; (2) select a rectangular neighborhood of the point; (3) segment the hand part from the patch using a threshold of difference of the depth value with respect to the hand point; (4) extract the contour of the hand using find_contours of openCV; (5) compute the maximum inscribed circle of the contour and take its center as the palm center; (6) remove additional contour points which belong to the wrist, and arrange the remaining contour points from the thumb to the little finger (i.e., in the clockwise/counterclockwise direction for the left/right hand).

3.2 Curvature scale space descriptors

The CSS descriptor is a contour-based shape descriptor derived from the zero-cross points of curvature of the curve convoluted with the Gaussian function. The greater the variance at which the curvature of the convoluted curve vanishes, the sharper the corresponding point of the curve is.

Let $(x(t), y(t))$ be a parametrization representation of a planar curve. The shape is evolved

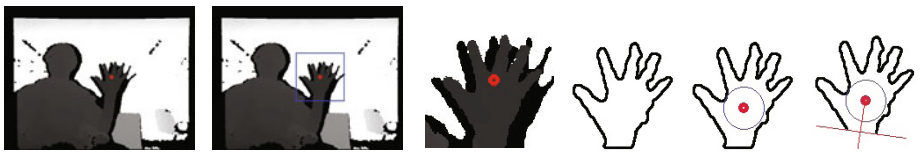

Fig. 1 A flowchart of extracting the hand contour and palm center: (a) finding a point on the hand; (b) selecting a rectangular neighborhood; (c) segmenting the hand part; (d) extracting the contour of the hand part; (e) computing the maximum inscribed circle of the contour; (f) removing additional contour points

into different scales by applying Gaussian smoothing $x_\sigma(t) = x(t) \otimes g(t,\sigma)$, $y_\sigma(t) = y(t) \otimes g(t,\sigma)$, where '$\otimes$' denotes the univariate convolution operator and $g(t,\sigma)$ is a Gaussian function. The curvature of the evolving curve is given by

$$k_\sigma(t) = [x'_\sigma(t)y''_\sigma(t) - x''_\sigma(t)y'_\sigma(t)] \cdot \left[(x'_\sigma(t))^2 + (y'_\sigma(t))^2\right]^{-3/2},$$

where $x'_\sigma(t), y'_\sigma(t), x''_\sigma(t), y''_\sigma(t)$ are the first and the second derivatives of $x_\sigma(t), y_\sigma(t)$ at location t, respectively. The CSS contour map

$$\mathrm{css}(t,\sigma) := \{(t,\sigma) : k_\sigma(t) = 0\} \quad (1)$$

is defined to be the collection of all zero-crossing points of (t,σ), where σ is the scale at which the curvature of the point t of the evolving curve vanishes.

3.3 Extracting fingertips and finger-valleys using CSS descriptors

3.3.1 Extracting fingertips

For each input hand contour $\{\boldsymbol{f}_t\}_{t=1}^n$, we compute its CSS contour map $\mathrm{css}_f(t,\sigma)$. Instead of directly using Eq. (1), we modify the traditional CSS contour map in two ways. First, since the obtained hand contour is a discrete sequence of points, the convolution operator and the curvature both need discretization forms (We use traditional discrete convolution operator for computing the convolution, and use quadratic fitting scheme for computing the curvature). Second, we define the CSS contour map by the absolute small value of the curvature of the evolving curve, instead of the zero points of the curvature of the curve directly. The reason is two-fold: for one thing, all the peak points of the hand contour do not exactly achieve zero curvature after the Gaussian smoothing (the first row of subfigures of Fig. 2); for another, setting the CSS contour as points of the curvature between zero and another small positive number leads to an extremely large number of points of CSS contours (the second row of subfigures of Fig. 2). Alternatively, our method defines a modified CSS contour map (Eq. (2)) by the collection of points which achieve the curvature within the interval $[c_1, c_2]$ of two small positive numbers (the third row of subfigures of Fig. 2), and extracts the fingertips using the local maxima of the contour with a

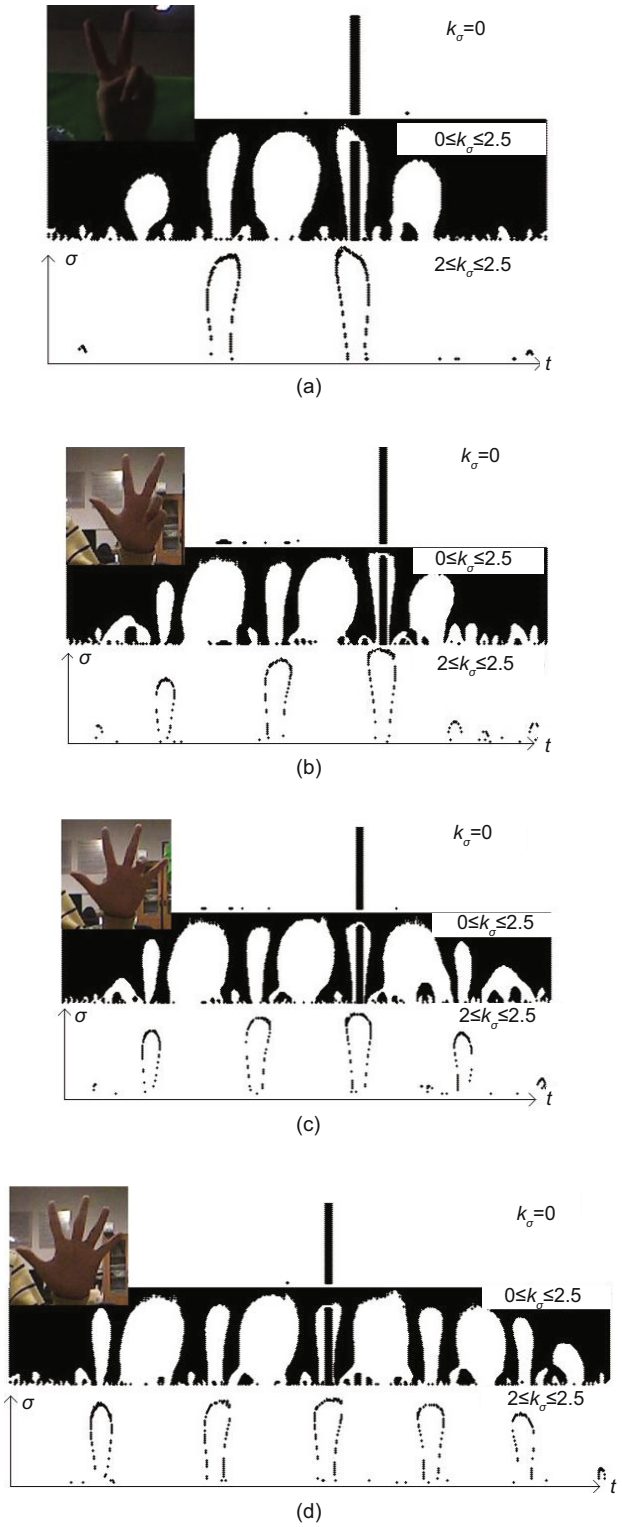

Fig. 2 Curvature scale space contours for two (a), three (b), four (c), and five (d) straight fingers in three cases: the first row satisfies $\mathrm{css}(t,\sigma) = \{(t,\sigma) : k_\sigma(t) = 0\}$, the second row satisfies $\mathrm{css}(t,\sigma) = \{(t,\sigma) : 0 \le k_\sigma(t) \le 2.5\}$, and the third row satisfies $\mathrm{css}(t,\sigma) = \{(t,\sigma) : 2 \le k_\sigma(t) \le 2.5\}$

threshold of σ.

$$\mathrm{css}_f^{\mathrm{tip}}(t,\sigma) := \{(t,\sigma) : 0 < c_1 \le k_\sigma(t) \le c_2\}, \quad (2)$$

$$\mathrm{css}_f^{\mathrm{valley}}(t,\sigma) := \{(t,\sigma) : c_3 \le k_\sigma(t) \le c_4 < 0\}. \quad (3)$$

3.3.2 Extracting finger-valleys

Next we extract finger-valleys. Although these may not be considered as hand articulations, they are important for computing finger-roots. We collect the points of curvature within an internal of two small negative numbers to obtain another CSS contour (Eq. (3)), and extract the finger-valleys using the local maxima of the contour.

Even if all fingers are straight, both the outer finger-valley of the thumb and the outer finger-valley of the little finger cannot be detected using CSS as both of them are smooth (When the thumb is bending, both the outer finger-valley of the first straight finger and the outer finger-valley of the last straight finger cannot be detected. We find them using the same method). The outer finger-valley of the thumb is added by the point which is symmetric to the first finger-valley with respect to the fingertip of the thumb. Similarly, the outer finger-valley of the little finger is added by the point symmetric to the last finger-valley with respect to the fingertip of the little finger.

3.3.3 Extracting finger-roots

Currently, we compute only a finger-root for each detected fingertip; when the fingertip is not detected by the CSS contour, the corresponding finger-root cannot be located, and we shall discuss this issue in Section 3.4. For each detected fingertip, the corresponding finger-root is simply given by the middle point of two neighboring finger-valleys of the fingertip.

3.4 Recovering missing fingertips and missing finger-roots

The CSS contour map can effectively detect straight or slightly bending fingers, but is ineffective for bending fingers. To recover bending fingers whose fingertips do not appear on the hand contour, we propose angle thresholds for recovering bending non-thumb fingertips. We first determine whether the thumb is bending according to the existence of the fingertip within the [15%]n-th$\sim$ [25%]n-th points of the whole hand contour, where n denotes the number of contour points ordered from thumb to little. If the thumb is bending, we roughly determine the finger-root of the thumb by the [20%]n-th point of the hand contour. Then we recover the fingertips of bending non-thumb and leave the recovery of the fingertip of the thumb at the end of this subsection (if it is missing). We connect each detected finger-root to the palm center and compute the angle of the ith non-thumb finger-root with respect to the finger-root of the thumb, denoted by θ_i, $1 \le i \le s$, where $s \le 4$ denotes the number of detected non-thumb fingers (Fig. 3). We also denote the following four intervals of the angles, each of which is associated with a non-thumb finger:

$$\Omega_i = \big[d_i + (45 - \mathrm{depth}(\mathbf{pc}))d_0, \\ d_{i+1} + (45 - \mathrm{depth}(\mathbf{pc}))d_0\big),\ i = 1,2,3,4,$$

where depth($\mathbf{pc}$) denotes the real depth value of the palm center, and d_i ($i = 0, 1, \ldots, 5$) are the parameters which shall be given in Section 4. The forefinger, middle finger, ring finger, and little finger can be judged missing and bending if $\Omega_i \cap \{\theta_j\}_{j=1}^{s} = \varnothing$ holds for $i = 1, 2, 3, 4$, respectively.

Once the ith non-thumb finger is bending, we select the ray $\boldsymbol{L}$ starting from the palm center at the middle angle (i.e., $(d_i+d_{i+1})/2+(45-\mathrm{depth}(\mathbf{pc}))d_0$) of the corresponding interval, and compute the intersections of $\boldsymbol{L}$ and the hand contour. The missing finger-root is determined by the nearest intersection to the palm center (Usually there is only one intersection between $\boldsymbol{L}$ and the contour. However, when $\boldsymbol{L}$ has a bias direction it may intersect with the contour points of other fingers. In this case the nearest point to the palm center is the correct finger-root). The missing fingertip is then determined by one of the points of the line segment between the finger-root and the palm center which achieves the greatest directional derivative of depths along the counter-direction of $\boldsymbol{L}$. We illustrate the whole procedure in the case of a straight thumb in Fig. 3.

Finally, we recover the fingertip of the thumb if the thumb is bending. If all the fingers are straight except the thumb (see the third row of Fig. 10), we extract the points whose depths are smaller than the depth of the palm center, and take the farthest point from the finger-root of the thumb as the

fingertip of the thumb. If the hand contains bending fingers other than the thumb (see the fourth to seventh rows of Fig. 10), we shall handle such a complicated case carefully. We collect all the points within the polygon whose vertices are five finger-roots and the first and the last point of the contour. Among this collection, we extract the points whose depth is smaller than the depth of the palm center. To separate the points of bending thumb from the points of bending non-thumb fingers, we remove the points whose depth equals the depth of any bending non-thumb fingertip and the points whose depth equals the depth of any bending non-thumb finger-root (We note that this removal involves all finger-tips and finger-roots of non-thumb fingers which are bending). Finally, the finger-root of the thumb is given by the farthest point from the finger-root of the thumb among this collection. We illustrate this procedure of recovering the fingertip of the bending thumb in Fig. 4. The whole procedure for both cases is given in Algorithm 1.

4 Experimental results

We give experimental results in this section. Experiment 1 uses both intensity images and depth images of human hands captured from Kinect, while Experiment 2 merely uses depth images provided by Microsoft Research Asia (http://research.microsoft.com/en-us/um/people/yichenw/handtracking). In Experiment 1, we compare our method with the K-cos method (Lee and Lee, 2011) and the convex-hull method (Nagarajan *et al.*, 2012); in Experiment 2, we compare our method with K-cos method, the convex-hull method, and the K-curvature method (Cerezo, 2012). Because the Kinect SDK outputs only three points (https://msdn.microsoft.com/en-us/library/dn799273.aspx) which are fewer than the points produced by our method and other appearance-based methods we compare with, we ignore the comparison with the Kinect SDK.

4.1 Parameter setting

The experiments are run on a Core(TM)2 Quad CPU Q9450 2.66 GHz machine with 4 GB RAM using Visual Studio 2010. The parameters used in Sections 3.3 and 3.4 are given as follows:

$$\sigma_1 = \frac{1}{2}\max\{\sigma : (t,\sigma) \in \mathrm{css}_f^{\mathrm{tip}}(t,\sigma)\},$$
$$\sigma_2 = \frac{1}{2}\max\{\sigma : (t,\sigma) \in \mathrm{css}_f^{\mathrm{valley}}(t,\sigma)\},$$
$$c_1 = 2,\ c_2 = 2.5,\ c_3 = -1.5,\ c_4 = -1,\ d_0 = 0.03,$$
$$d_1 = 0.8,\ d_2 = 1.4,\ d_3 = 1.7,\ d_4 = 2.0,\ d_5 = 2.6.$$

We explain the choice of parameters as follows: The parameters c_i $(i = 1, 2, 3, 4)$ are determined by evolving hand contours with different regions of k_σ, and d_i $(i = 0, 1, \ldots, 5)$ are determined by testing a standard hand with five straight fingers from distance of 30 cm to distance of 60 cm. Readers may doubt the robustness of our system since it involves many artificial parameters. We argue that, because both the positions of curvature peak points and the bounds of angles between two adjacent fingers remain unchanged for different hands, the selection of those parameters leads to a robust system. The only issue is the long-time off-line testing.

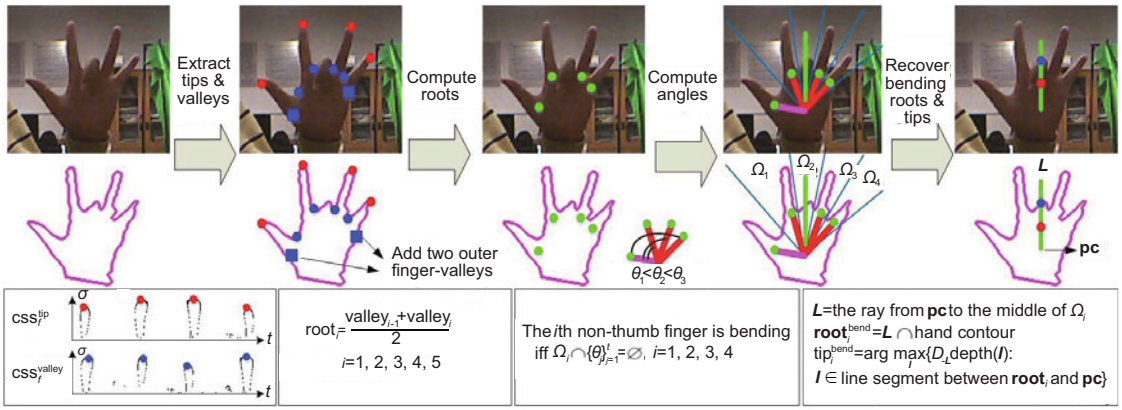

Fig. 3 A flowchart for detecting fingertips and finger-valleys and recovering bending non-thumb fingertips

Algorithm 1 Detecting hand articulations from the hand contour

Input: Hand contour $\{\boldsymbol{f}_t\}_{t=1}^{n}$, palm center $\mathbf{pc}$, and CSS parameters $\sigma_1, \sigma_2, c_1, c_2, c_3, c_4, d_0, d_1, d_2, d_3, d_4$
Output: The fingertips and finger-roots of the hand
Compute two CSS contours for fingertips and finger-valleys using Eqs. (2) and (3), respectively
Remove the points from $\mathrm{css}_f^{\mathrm{tip}}(t,\sigma)$ whose σ's value is greater than σ_1
Remove the points from $\mathrm{css}_f^{\mathrm{valley}}(t,\sigma)$ whose σ's value is greater than σ_2
Extract all the local maxima of $\mathrm{css}_f^{\mathrm{tip}}(t,\sigma)$ as fingertips $\{\mathbf{tip}_i\}_{i=1}^{s}$ // s: the number of detected non-thumb fingers
Extract all the local maxima of $\mathrm{css}_f^{\mathrm{valley}}(t,\sigma)$ as finger-valleys $\{\mathbf{valley}_i\}_{i=1}^{s-1}$
Add the outer finger-valley of the thumb and the little by: $\mathrm{order}(\mathbf{valley}_0) = 2\cdot\mathrm{order}(\mathbf{tip}_1) - \mathrm{order}(\mathbf{valley}_1)$, $\mathrm{order}(\mathbf{valley}_s) = 2\cdot\mathrm{order}(\mathbf{tip}_s) - \mathrm{order}(\mathbf{valley}_{s-1})$ // $\mathrm{order}(\boldsymbol{v})$: the index of a point $\boldsymbol{v}$ of $\{\boldsymbol{f}_t\}_{t=1}^{n}$
Determine the position of the finger-root of detected fingers as the middle point of two adjacent finger-valleys: $\mathbf{root}_i = (\mathbf{valley}_{i-1} + \mathbf{valley}_i)/2,\ i = 1, 2, \ldots, s$
if $\{\boldsymbol{f}_t\}_{15\%n\le t\le 25\%n} \cap \{\mathbf{tip}_i\}_{i=1}^{s} = \varnothing$ **then**
    $\mathbf{root}_{\mathrm{thumb}} \leftarrow f([20\%n])$ // when the thumb is undetected, the finger-root of the thumb is defined to be the // $[20\%]n$-th point of the contour
end if
if $s \le 3$ **then**
    // when non-thumb fingers are undetected, we use angle thresholds for recovering missing fingers
    $\theta_i \leftarrow \angle\mathbf{root}_i - \mathbf{pc} - \mathbf{root}_{\mathrm{thumb}},\ i = 1, 2, \ldots, s$ // compute angle θ_i between the ith non-thumb finger-root // and the finger-root of the thumb with respect to the palm center
    Compute four intervals Ω_i $(i = 1, 2, 3, 4)$ of angles associated with the four non-thumb fingers
    if $\Omega_i \cap \{\theta_j\}_{j=1}^{s} = \varnothing$ holds for some i **then**
        // the ith non-thumb finger is undetected if the ith interval contains no angles of $\{\theta_j\}_{j=1}^{s}$
        $\boldsymbol{L} \leftarrow$ the ray starting from $\mathbf{pc}$ to the middle of Ω_i
        $\mathbf{root}_i \leftarrow \arg\min_{\boldsymbol{l}} \{\|\boldsymbol{l} - \mathbf{pc}\| : \boldsymbol{l} \in \boldsymbol{L} \cap \{\boldsymbol{f}_t\}_{t=1}^{n}\}$ // the missing finger-root is determined by one of the intersections // of the ray $\boldsymbol{L}$ and the hand contour which is nearest to the palm center
        $\mathbf{tip}_i \leftarrow \arg\max_{\boldsymbol{l}} \{D_{-\boldsymbol{L}}\mathrm{depth}(\boldsymbol{l}) : \boldsymbol{l} \in$ the line segment between $\mathbf{root}_i$ and $\mathbf{pc}\}$ // the missing fingertip is // determined by one of the points of the line segment between the finger-root and the palm center which achieves // the greatest directional derivative of depths along the counter-direction of the ray $\boldsymbol{L}$, denoted by $D_{-\boldsymbol{L}}\mathrm{depth}(\boldsymbol{l})$
    end if
end if
if $\mathbf{tip}_{\mathrm{thumb}}$ is undetected **then**
    Denote Θ_0 to be the heptagon whose vertices consist of all finger-roots and the first and last contour points
    $\Theta_1 \leftarrow \{\boldsymbol{v} \in \Theta_0 : \mathrm{depth}(\boldsymbol{v}) < \mathrm{depth}(\mathbf{pc})\}$ // see Fig. 4
    if $s = 4$ **then**
        $\mathbf{tip}_{\mathrm{thumb}} \leftarrow \arg\max_{\boldsymbol{l}\in\Theta_1} \|\boldsymbol{l} - \mathbf{root}_{\mathrm{thumb}}\|$ // when all non-thumb fingers are straight, we select the farthest // point of Θ_1 from the finger-root of the thumb as the fingertip of the thumb
    else
        $\Theta_2 \leftarrow \{\boldsymbol{v} \in \Theta_0 : \mathrm{depth}(\boldsymbol{v}) = \mathrm{depth}(\mathbf{root}_i)$ or $\mathrm{depth}(\boldsymbol{v}) = \mathrm{depth}(\mathbf{tip}_i), 1 \le i \le s\}$
        $\mathbf{tip}_{\mathrm{thumb}} \leftarrow \arg\max_{\boldsymbol{l}\in\Theta_1\setminus\Theta_2} \|\boldsymbol{l} - \mathbf{root}_{\mathrm{thumb}}\|$ // when some non-thumb fingers are bending, we remove the // points belonging to the bending fingers and then select the farthest point as the fingertip of the thumb
    end if
end if

4.2 Experiment 1: Kinect data

4.2.1 Quantitative results

Experiment 1 takes 21 depth images with various poses of hands captured by Kinect. We compute the pixel-wise distance between the ground truth which we mark manually and the location of each detected fingertip obtained by using our method, the K-cos method (Lee and Lee, 2011), and the convex-hull method (Nagarajan *et al.*, 2012). In particular, we compute the distance for each detected fingertip only using the CSS phase. Both the root-of-mean-square (RMS) and the maximum of the errors of all test images for each fingertip are shown in Fig. 5.

The results indicate that the CSS phase achieves the smallest pixel-wise error while the whole procedure of our method achieves the greatest. This is because our method adds bending fingertips by a rough estimation, but that does not imply that our method is less effective than the K-cos and the convex-hull methods. To argue this, we count the number of undetected fingertips and incorrectly detected fingertips of the CSS phase and show the results in Fig. 6. While the whole procedure of our method produces no missing fingertips and no incorrect fingertips, the CSS phase produces one incorrect fingertip while the other two methods produce over 40 incorrect ones.

4.2.2 Qualitative results

To show the qualitative results of Experiment 1 in a vivid fashion, we model a kinematic hand configuration with 60 articulation parameters, i.e., three-dimensional coordinates of five fingertips, five finger-roots, the palm center, and nine finger joints. The x, y-coordinates are given by the locations of hand articulations while the z-coordinate is given by the depth value of the corresponding location. All the fingers contain two joints except the thumb which contains one. The joints are computed equidistantly using the coordinates of fingertips and finger-roots. The hand model consists of fourteen cylinders and a plane, where each cylinder is determined by a finger joint and a fingertip (or a finger-root), and the plane fits to all the finger-roots and the palm center (Fig. 7).

We show the qualitative results of Experiment 1 in Figs. 8–10. We mark the fingertips and

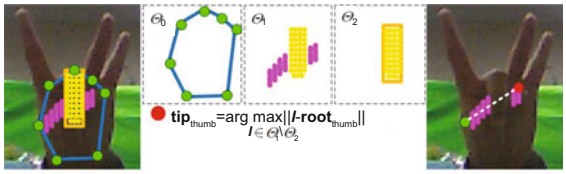

Fig. 4 Recovery of the undetected fingertip of the thumb when the thumb is bending

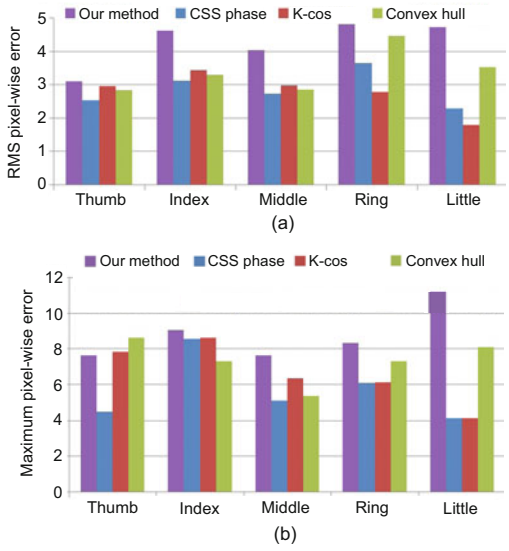

Fig. 5 The RMS error (a) and the maximum error (b) of all test images for each fingertip

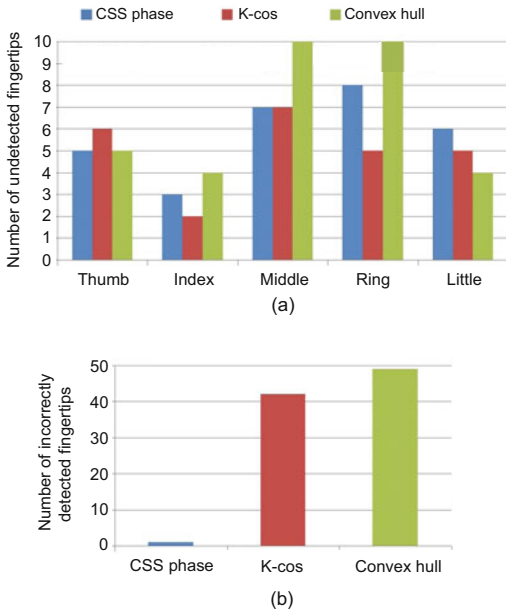

Fig. 6 The number of undetected fingertips (a) and the number of incorrectly detected fingertips (b) among all test images

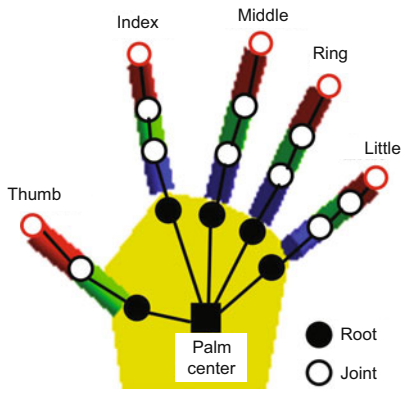

Fig. 7 The hand model consists of fourteen cylinders and a plane, characterized by the three-dimensional coordinates of a palm center, five finger-roots, nine finger joints, and five fingertips

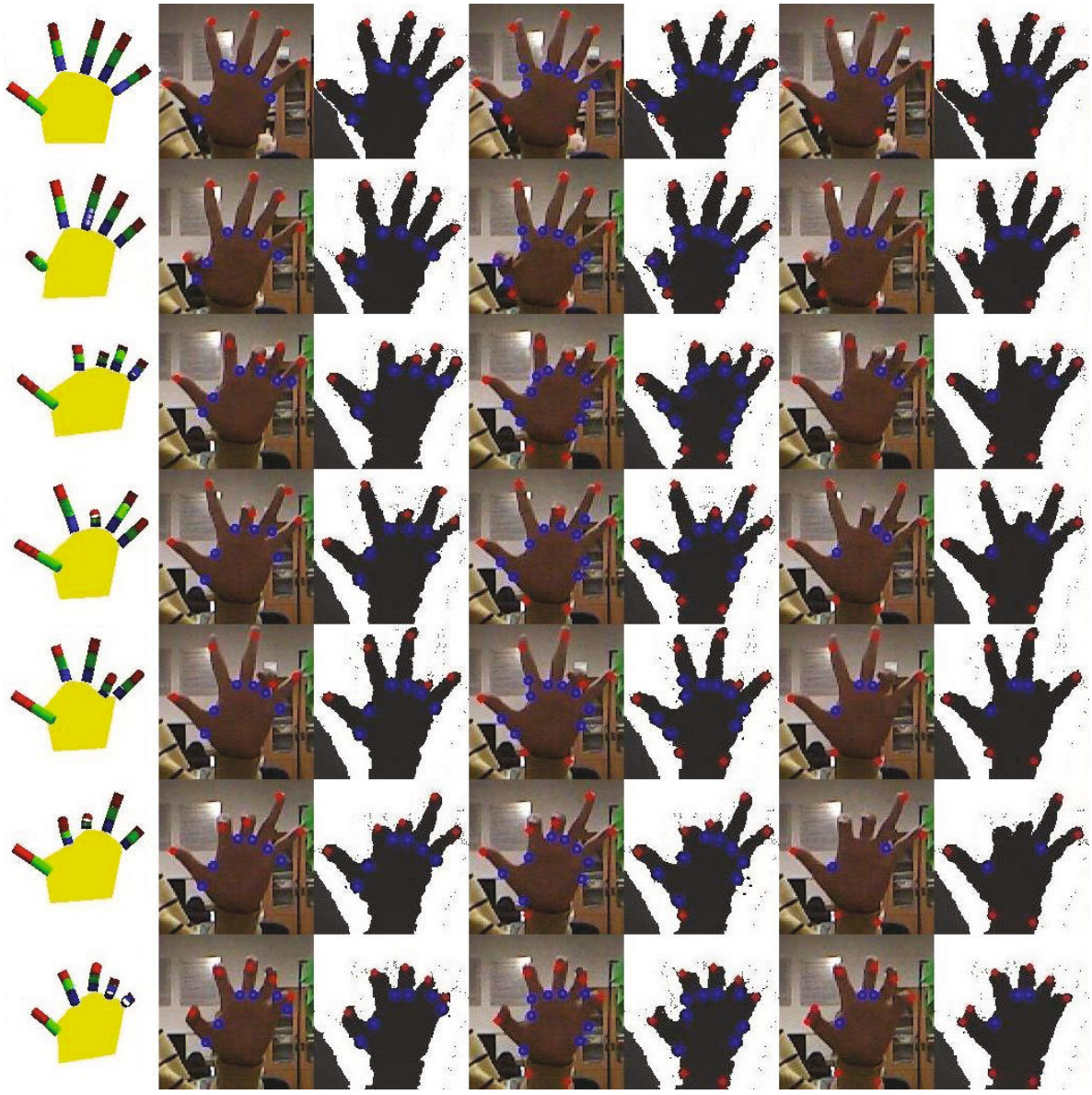

Fig. 8 Qualitative results of hand articulations of Experiment 1 (red: fingertips; blue: finger-valleys): (a) hand models generated by our method; (b) and (c) the CSS phase; (d) and (e) the K-cos method (Lee and Lee, 2011); (f) and (g) the convex-hull method (Nagarajan *et al.*, 2012). All the fingertips and finger-valleys are detected by CSS contours. References to color refer to the online version of this figure

finger-valleys obtained by three methods on both color images and depth images, and we show the final model obtained by our method. We see that, the examples of Fig. 8 capture all fingertips and finger-valleys using only the CSS phase. However, the CSS phase cannot capture all fingertips or finger-valleys in the examples of Figs. 9 and 10. This is because those examples contain bending fingers whose fingertips do not locate at the hand contour. Fortunately, the phase of recovering bending fingertips works for those examples. The third and fourth rows show a good recovery of locations of bending fingertips. However, when the thumb is occluded or when more than two fingers are bending as in the fourth to seventh rows of Fig. 10, the detected locations lack accuracy. In the fifth row, the fingertip of the thumb is actually occluded by the ring finger while our method recovers the fingertip of the thumb in front of the ring

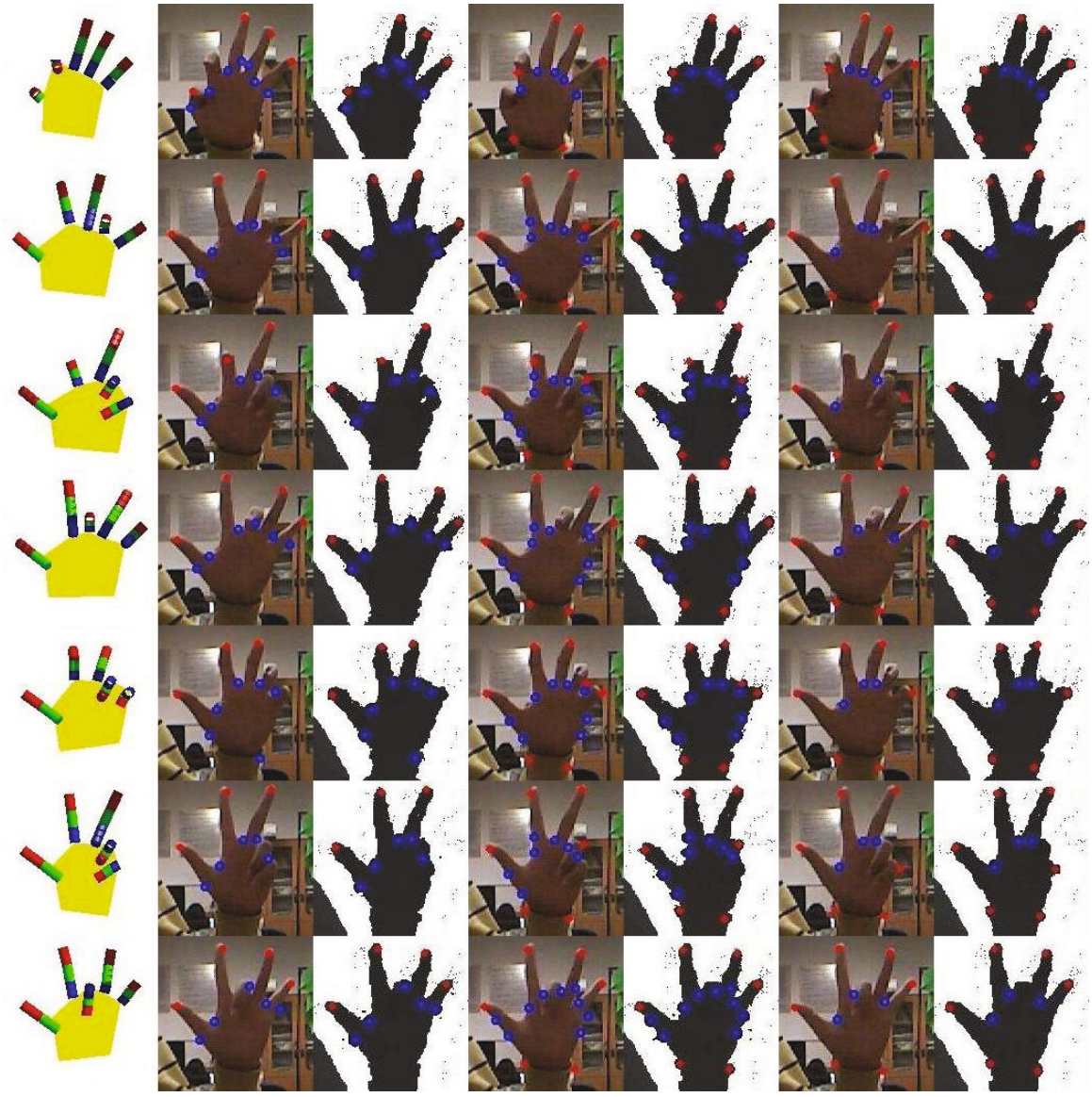

Fig. 9 Qualitative results of hand articulations of Experiment 1 (red: fingertips; blue: finger-valleys): (a) hand models generated by our method; (b) and (c) the CSS phase; (d) and (e) the K-cos method (Lee and Lee, 2011); (f) and (g) the convex-hull method (Nagarajan *et al.*, 2012). One or two fingers are bending. References to color refer to the online version of this figure

finger. In particular, the sixth and seventh rows of Fig. 10 indicate that our method cannot effectively handle highly occlusive cases.

4.3 Experiment 2: MSRA database

We show the qualitative results of Experiment 2 in Fig. 11. While the CSS method works much better than the K-cos and convex-hull methods, the K-curvature method also provides a few good results, except for the 2nd, 12th rows on the left and the 1st, 8th, 10th rows on the right. We argue that the K-curvature method is more robust than the previous two appearance-based methods in boundary detection (especially when the hand contour cannot be clearly captured), but treats occlusion cases ineffectively. In addition, the K-curvature cannot detect finger-valley points and cannot help find finger joints.

Fig. 12 shows some failure examples of Experiment 2 using the CSS method. We consider those

Fig. 10 Qualitative results of hand articulations of Experiment 1 (red: fingertips; blue: finger-valleys): (a) hand models generated by our method; (b) and (c) the CSS phase; (d) and (e) the K-cos method (Lee and Lee, 2011); (f) and (g) the convex-hull method (Nagarajan *et al.*, 2012). Three or four fingers are bending, or the thumb is bending. References to color refer to the online version of this figure

examples as failures since our method misses or incorrectly detects at least two fingertips of the target hand. In general, our method fails for the examples with large occlusions or with small resolutions. Such a disadvantage is common for appearance-based methods and can be improved by model-based methods. This is the future work we shall consider.

4.4 Computational time

The average running times for our method, the K-cos method (Lee and Lee, 2011), the convex-hull method (Nagarajan *et al.*, 2012), and the K-curvature method (Cerezo, 2012) are 1.96 825 s, 0.648 467 s, 0.755 888 s, and 0.02 s, respectively. The

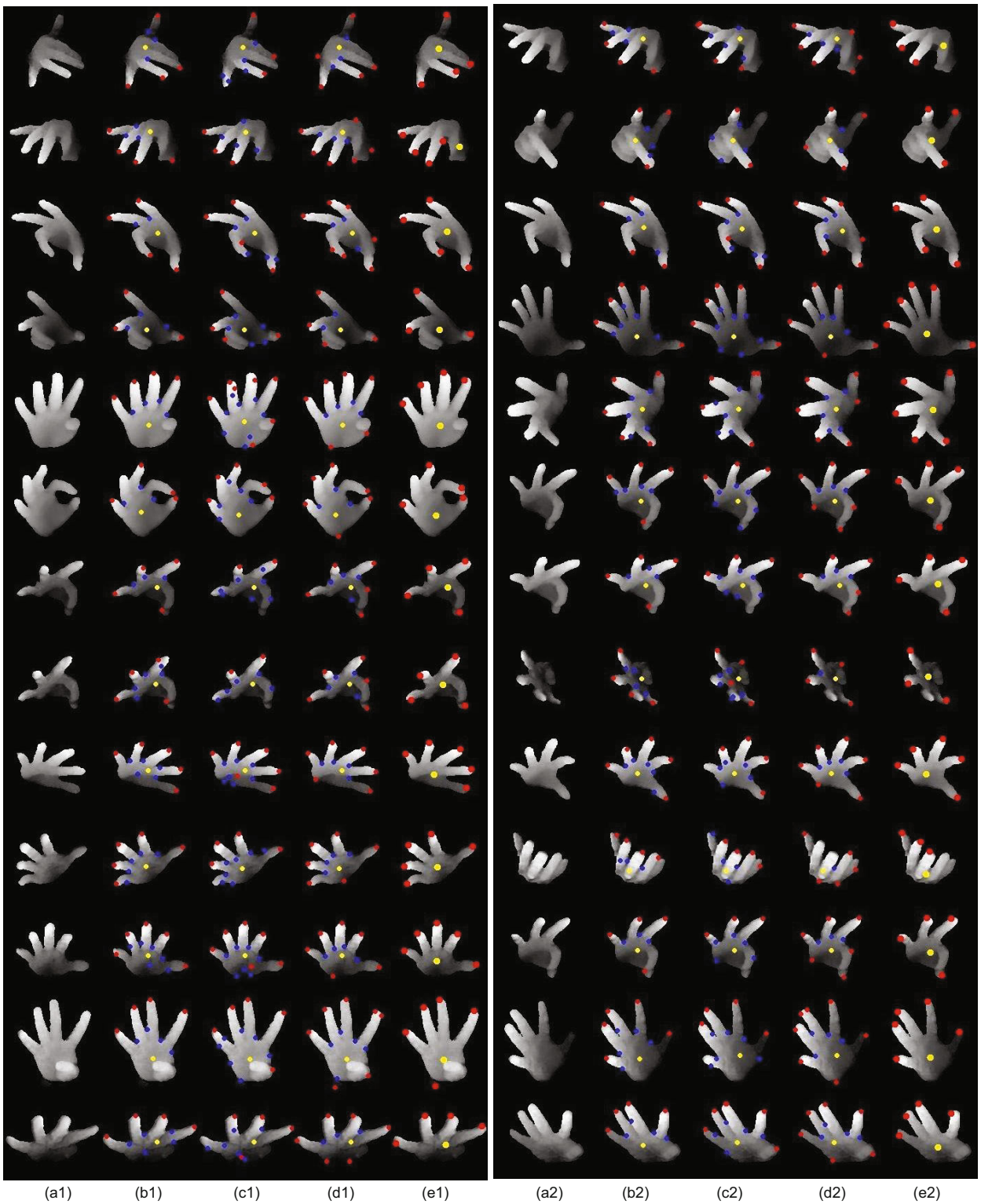

Fig. 11 Qualitative results of hand articulations of Experiment 2 (red: fingertips; blue: finger-valleys; yellow: palm center): (a1) and (a2) are the original depth images, (b1) and (b2) the CSS method, (c1) and (c2) the K-cos method (Lee and Lee, 2011), (d1) and (d2) the convex-hull method (Nagarajan *et al.*, 2012), (e1) and (e2) the K-curvature method (Cerezo, 2012). References to color refer to the online version of this figure

CSS method is slower than the others because our method spends a lot of time obtaining the CSS contours, and this involves a quadratic fitting step. More efficient discretization schemes of curvature of the contour will improve our method.

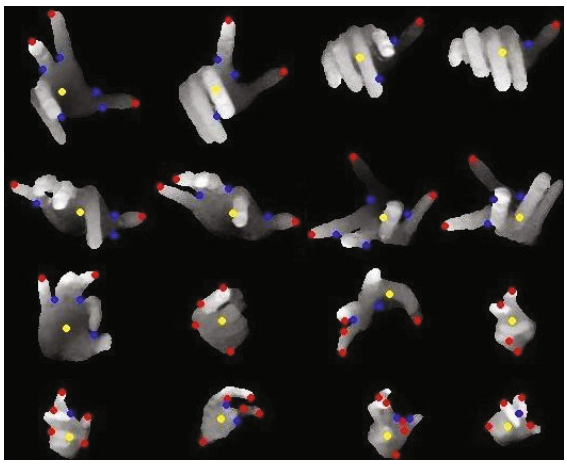

Fig. 12 Failure examples of the CSS method of Experiment 2 (red: fingertips; blue: finger-valleys; yellow: palm center). References to color refer to the online version of this figure

5 Conclusions

We have proposed an appearance-based method for extracting both straight fingers and bending fingers, using a modified CSS descriptor and the angle thresholds characterized by the finger-root of the thumb and the palm center. Experimental results showed that our method detects hand articulations more precisely than three state-of-the-art appearance-based approaches. This is because the CSS descriptors are more robust for noisy or corrupted data. When input images contain two or three bending fingers, our method can detect them with a low accuracy. This is because 2D curvature or other geometric descriptors are difficult to handle in such highly occlusive cases.

Our method has the following disadvantages. (1) We can handle only hand images with normalized orientation. For abnormalized orientation of depth images, our method will fail in detecting bending fingers because the thresholds of depth values no longer work. (2) When images contain many bending fingers, the 2D CSS descriptor may recognize finger joints as fingertips incorrectly as the joints appear on the 2D hand contour. (3) Too many parameters require a long-time and extensive off-line testing.

To improve our method, we shall consider extracting 3D CSS contours for 3D hand contours (i.e., the 3D curves around five fingers including bending fingers) and integrating the 3D CSS descriptors with a parametric model of human hands, which might better treat bending fingers.

References

Abbasi, S., Mokhtarian, F., Kittler, J., 1999. Curvature scale space image in shape similarity retrieval. *Multimedia Syst.*, **7**(6):467-476. http://dx.doi.org/10.1007/s005300050147

Athitsos, V., Sclaroff, S., 2002. An appearance-based framework for 3D hand shape classification and camera viewpoint estimation. Proc. 5th IEEE Int. Conf. on Automatic Face and Gesture Recognition, p.40-45. http://dx.doi.org/10.1109/AFGR.2002.1004129

Athitsos, V., Sclaroff, S., 2003. Estimating 3D hand pose from a cluttered image. Proc. IEEE Computer Society Conf. on Computer Vision and Pattern Recognition, p.432-439. http://dx.doi.org/10.1109/CVPR.2003.1211500

Cerezo, T., 2012. 3D hand and finger recognition using Kinect. Technical Report, Universidad de Granada, Spain. Available at http://frantracerkinectft.codeplex.com.

Chang, W.Y., Chen, C.S., Jian, Y.D., 2008. Visual tracking in high-dimensional state space by appearance-guided particle filtering. *IEEE Trans. Image Process.*, **17**(7):1054-1067. http://dx.doi.org/10.1109/TIP.2008.924283

de La Gorce, M., Fleet, D.J., Paragios, N., 2011. Model-based 3D hand pose estimation from monocular video. *IEEE Trans. Patt. Anal. Mach. Intell.*, **33**(9):1793-1805. http://dx.doi.org/10.1109/TPAMI.2011.33

Feng, Z., Yang, B., Chen, Y., *et al.*, 2011. Features extraction from hand images based on new detection operators. *Patt. Recog.*, **44**(5):1089-1105. http://dx.doi.org/10.1016/j.patcog.2010.08.007

Keskin, C., Kıraç, F., Kara, Y.E., *et al.*, 2011. Real time hand pose estimation using depth sensors. *In*: Fossati, A., Gall, J., Grabner, H., *et al.* (Eds.), Consumer Depth Cameras for Computer Vision, Springer, London, p.119-137. http://dx.doi.org/10.1007/978-1-4471-4640-7_7

Kirac, F., Kara, Y.E., Akarun, L., 2014. Hierarchically constrained 3D hand pose estimation using regression forests from single frame depth data. *Patt. Recog. Lett.*, **50**:91-100. http://dx.doi.org/10.1016/j.patrec.2013.09.003

Lee, D., Lee, S., 2011. Vision-based finger action recognition by angle detection and contour analysis. *ETRI J.*, **33**(3):415-422. http://dx.doi.org/10.4218/etrij.11.0110.0313

Ma, Z., Wu, E., 2014. Real-time and robust hand tracking with a single depth camera. *Vis. Comput.*, **30**(10):1133-1144. http://dx.doi.org/10.1007/s00371-013-0894-1

Maisto, M., Panella, M., Liparulo, L., *et al.*, 2013. An accurate algorithm for the identification of fingertips using an RGB-D camera. *IEEE J. Emerg. Sel. Topics Circ. Syst.*, **3**(2):272-283. http://dx.doi.org/10.1109/JETCAS.2013.2256830

Morshidi, M., Tjahjadi, T., 2014. Gravity optimised particle filter for hand tracking. *Patt. Recog.*, **47**(1):194-207. http://dx.doi.org/10.1016/j.patcog.2013.06.032

Nagarajan, S., Subashini, T., Ramalingam, V., 2012. Vision based real time finger counter for hand gesture recognition. *Int. J. Technol.*, **2**(2):1-5.

Oikonomidis, I., Kyriazis, N., Argyros, A.A., 2011. Efficient model-based 3D tracking of hand articulations using Kinect. *BMVC*, **1**(2):1-11.

Qian, C., Sun, X., Wei, Y., *et al.*, 2014. Realtime and robust hand tracking from depth. Proc. IEEE Conf. on Computer Vision and Pattern Recognition, p.1106-1113. http://dx.doi.org/10.1109/CVPR.2014.145

Ren, Z., Yuan, J., Zhang, Z., 2011. Robust hand gesture recognition based on finger-earth mover's distance with a commodity depth camera. Proc. 19th ACM Int. Conf. on Multimedia, p.1093-1096. http://dx.doi.org/10.1145/2072298.2071946

Rosales, R., Athitsos, V., Sigal, L., *et al.*, 2001. 3D hand pose reconstruction using specialized mappings. Proc. 8th IEEE Int. Conf. on Computer Vision, p.378-385. http://dx.doi.org/10.1109/ICCV.2001.937543

Schlattmann, M., Kahlesz, F., Sarlette, R., *et al.*, 2007. Markerless 4 gestures 6 DOF real-time visual tracking of the human hand with automatic initialization. *Comput. Graph. Forum*, **26**(3):467-476. http://dx.doi.org/10.1111/j.1467-8659.2007.01069.x

Tomasi, C., Petrov, S., Sastry, A., 2003. 3D tracking = classification + interpolation. Proc. 9th IEEE Int. Conf. on Computer Vision, p.1441-1448. http://dx.doi.org/10.1109/ICCV.2003.1238659

Tompson, J., Stein, M., Lecun, Y., *et al.*, 2014. Real-time continuous pose recovery of human hands using convolutional networks. *ACM Trans. Graph.*, **33**(5):169.1-169.10. http://dx.doi.org/10.1145/2629500