EURASIP Journal on Embedded Systems
a SpringerOpen Journal

RESEARCH Open Access

A methodology for hand and finger motion analysis using adaptive probabilistic models

Chutisant Kerdvibulvech

Abstract

A methodology for motion analysis and hand tracking based on adaptive probabilistic models is presented. This is done by integrating a deterministic clustering framework and a particle filter together in real time. The skin color of a human hand is firstly segmented. A Bayesian classifier and an adaptive process are utilized for determining skin color probabilities. The methodology enables us to deal with luminance changes. After that, we determine the probabilities of the fingertips by using semicircle models for fitting curves to fingertips. Following this, the deterministic clustering algorithm is utilized to search for regions of interest, and then the Sequential Monte Carlo is also performed to track the fingertips efficiently. Representative experimental results are also included to ensure workability of the proposed framework. Several issues about using the presented method in embedded systems are discussed. The method presented can be used to further develop the associated applications of embedded robotic and virtual reality.

Keywords: Motion analysis; Hand tracking; Extended sequential Monte Carlo; Finger tracking; Color segmentation; Bayesian classifier; Embedded system; Adaptive learning; Clustering algorithm

1 Introduction

Recently, embedded systems are beneficially applied to many autonomous and intelligent robotic fields. One of the possible keys is to make the embedded robot see and understand automatically. In many embedded systems, vision-based methods are used interestingly. Their algorithms are embedded in robots in both hardware and software, including a method about hand motion analysis. This is because if embedded robotic systems are able to recognize human organs automatically, they can apply to various related real-life applications practically. An example includes embedded robots used and researched after 9/11 which are designed to automatically operate and rescue humans within a challenging environment by recognizing human organs without using human eyes. Thus, it is very important to design the embedded robots that can recognize and analyze the motion of human organs in recent years. For this reason, researches about hand motion recognition based on digital image processing technology are becoming popular for embedded systems. This is because computer vision has been applied to many kinds of recent application to assist human motion tracking, especially fingertip tracking methodologies. Previous fingertip tracking methods were presented. For example, a correlation with pre-defined templates was presented in [1]. A chromatic distance was discussed in [2]. Mackie and McCane [3] also proposed image-division-based decision tree recognition. However, these aforementioned methods are not directly applicable to the self-occlusion fingertip tracking. Moreover, the background they used is usually uniform. As a result, it is more complicated to locate the fingertip positions correctly for self-occlusion and in non-uniform background. The proposed methodology for tracking the hand and fingertips solves these aforementioned issues.

To begin with, the hand is segmented in each frame from the background using an adaptive color detection algorithm. A Bayesian classifier is utilized during off-line phase [4]. An adaptive algorithm for determining skin probability is then applied to refine the classifier to train the system robustly [5]. Following this, we determine probabilities for fingertips by cropping the models of semicircle shape for a fit to the fingertip [6]. After superimposing the models on every candidate in the test

Correspondence: chutisant.k@rsu.ac.th
Department of Information and Communication Technology, Rangsit University, 52/347 Muang-Ake, Paholyothin Road, Lak-Hok, Patum Thani 12000, Thailand

image, we normalize the results which will be used as the fingertip probability map for tracking. Next, a clustering approach [7] is used to determine for regions of interest (ROIs) and sequential Monte Carlo [8] method is used for tracking by distributing the particles inside the corresponding ROIs. This vision-based methodology enables us to track the fingertips even when some fingers are not fully stretched out or when the luminance changes.

This paper is structured as follows. Literature on previous and conventional works is reviewed in Section 2. Next, the series of steps presented for hand and finger motion analysis using adaptive probabilistic models is described in Section 3. After that, Section 4 provides the experimental setup, including the results and discussion. Ultimately, Section 5 gives a summary of the paper and discusses possible associated embedded robotic applications using the proposed vision-based method.

2 Related work

Previous works about gesture recognition have been shown useful for various applications. Martínez et al. [9] developed a system for sign language to recognize motion primitives and full sentences. They assume that the same sign has different meanings depending on context. Matilainen et al. [10] presented a finger tracking system using template matching for gesture recognition, focusing on mobile devices. Krejov and Bowden [11] presented a system using a weighted graph and depth information of the hand for determining the geodesic maxima of the surface. In [12], Kereliuk et al. detected the positions of fingertips. The circular Hough transform is used for determining the tips of the fingers.

Nevertheless, these aforementioned gesture recognition methods are not suitably applicable to the fingertip tracking when self-occlusion occurs. Also in [11], the hand and wrist localization works not so smoothly and robustly, while from our experiments, utilizing the Hough transform to detect the fingertips in [12] is not robust enough. This is because fingertip edges cannot be easily detected due to the noise around the fingertips. Also, they did not aim to deal with luminance changes in online process.

We overcome these problems by attempting to segment the skin color of hand robustly. To solve this issue, it is important to understandably address a problem to control the lighting [13]. The levels of light between off-line and online phases are important for getting the correct registration. A major decision has to be made when deriving a model of color. By simply setting the threshold in color model, the accurate and robust results are rarely obtained. Another method [14] is to use histogram models. Still, it cannot perform adaptively when the levels of light between off-line and online phases are totally different.

To solve this issue, a Bayesian classifier is utilized. Applying this method, the first advantage is that the system can automatically and adaptively learn the probabilities by itself during online phase. From a small amount of training data, the probability is adapted during online phase and converges automatically to a proper value. Thus, it allows us to segment the regions we need robustly even though changing of luminance happens.

3 Methods

The schematic of the implementation will be explained in this section. After capturing the images, a Bayesian classifier is utilized adaptively to segment the human hand. As the next step, we apply a matching algorithm to determine the probabilities of the fingertips (i.e., fingertip probability map). Then, we extend the standard particle filter by utilizing the clustering algorithm to create ROIs for tracking. In this way, the positions of human hand and fingertips can be visually tracked.

3.1 Hand region segmentation

If the projection matrix is known, we can calculate the homography for warping a pre-captured known background. However, the background we used is sometimes dynamic and the background for that area cannot be easily synthesized. The luminance changing also causes a problem for using pre-captured known background image. The pixel color of pre-captured known background and the one from current input would be very different.

In our approach, we want to segment a hand from the input image. We built a color model of the hand image. During learning phase, the color model is also adapted according to changing luminance. In other words, we assume that the hand is a known foreground color model.

To begin with, we calculated the color probabilities being skin color by applying [4] which is composed of two main phases: off-line phase and online phase. First, we selected some images to train the system manually. Second, the probability is updated automatically and adaptively from the new input images [5]. In our implementation, we set that the adapting process is automatically disabled as soon as the probabilities are stable. Hence, when we start to learn the online skin color adaptation, we assume that there is enough skin in the image. As soon as the online adapting process is enough as we prefer (i.e., the skin color probability converges to a proper value), we manually stop the adapting process. In this way, after finishing the online learning process, though the skin area disappears from the scene, it does not affect the skin color probability.

3.1.1 Off-line phase

Wei et al. [15] suggested that skin color model based on this space for object segmentation and classification

using 3D range camera can provide interesting coverage of human in many races. Similarly, we use their assumption for a 3D color representation (YUV). Nevertheless, we use only UV as it demands less memory storage. This disregard of the luminance value has also been shown to be useful in detection and tracking of color night vision [16]. During an off-line phase, Bayes' rule is used for estimating the probability $P(s|c)$ of a color, with c being a skin color using

$$P(s|c) = \frac{P(c|s)P(s)}{P(c)} \quad (1)$$

where $P(s)$ is the proportion of the trained skin-colored pixels during off-line phase to the total number of pixels of whole images, $P(c)$ is the proportion of the number of occurrences of each color c to the total number of image points during training, and $P(c|s)$ is the proportion of the number of occurrences of a color c within the skin-colored regions to the number of skin-colored image points during training. After that, we use depth-first search method (DFS) to assign non-similar labels to the image pixels of non-similar regions. Filtering based on size of found regions is used to remove noise. Hence, connected components that consist of less than the threshold size are assumed to be noise and then rejected from further consideration. The threshold size for size filtering we used is 500 pixels. It is important to note that we do not need the intrinsic and extrinsic parameters in this step, since we assume that if the noise is smaller than the value we set, we simply eliminate it.

3.1.2 Online phase

This phase is similar to the off-line phase. We recalculate the probabilities again, but we use the values from the new input images. During an online phase, we update the adapted probabilities according to

$$P_A(s|c) = \gamma P(s|c) + (1-\gamma)P_W(s|c) \quad (2)$$

where $P_A(s|c)$ is the probability adapted of a color c being a skin color, γ is a sensitivity parameter, and W is the number of history frames. If W value is too high, the length of history frames will be too long; if W value is set too low, the history for adaptation will be too short. Figure 1 shows an example of skin segmentation by adaptive learning robustly. Using this adaptive framework, it is able to deal well with obvious luminance changes.

3.2 Determining the probabilities of tips

After segmenting the hand region, we use the semicircle models for a fit to the curved fingertip [6]. Six models are utilized to deal with different sizes and orientations of the tips of the fingers. We match semicircle templates against the results of hand segmentation by using

$$R(x,y) = \frac{\sum_{x',y'} [T(x',y') - H(x+x',y+y')]^2}{\sqrt{\left[\sum_{x',y'} \left(T(x',y')^2\right) \sum_{x',y'} H(x+x',y+y')^2\right]}} \quad (3)$$

where $T(x,y)$ is a searched template at coordinates (x,y), and $H(x,y)$ is a hand segmentation result when the search is running. Following this, we summarize the results of the fingertip models using $R_{\text{sum}}(x,y) = \sum_{i=1}^{N_0} R_i(x,y)$ where N_0 is a number of fingertip models.

Our experimental results have revealed that using the sum of the matches of all fingertip models gives the better result than other combinations (such as using maximum of the matches). A possible reason is that every model is weighted so that the information of all matches is used. In the case that if any matches of all fingertip models are not close to the answer (but the mean of the matches is close to the answer), this can still produce the promising result. However, using the maximum of the matches would give the good result if the results of the matches are very scattered, but from our experiments, this case rarely happens when matching the models for tracking the fingertips.

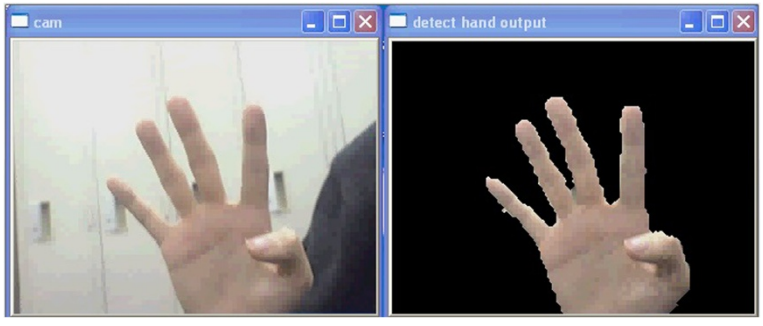

Figure 1 Skin segmentation by adaptive learning.

The models are then superimposed on every candidate during testing. Next, we normalize results of each model by using

$$R_{\text{normalized}}(x,y) = \frac{R_{\text{sum}}(x,y)}{R_{\text{sum}}(x_{\max},,y_{\max})} \quad (4)$$

where $\forall_{(x,y)}\{R_{\text{sum}}(x_{\max},y_{\max}) \geq R_{\text{sum}}(x,y)\}$. As a result, the probabilities to the fingertips of each pixel can be obtained.

3.3 Multiple fingertip tracking

Our method takes the advantages of sequential Monte Carlo [8] about automatic track initialization and recovering whenever the tracking fails. When the fingertips disappear from the scene and then appear back, we can still track the fingertips correctly due to the advantage of utilizing particle filter. However, direct application of sequential Monte Carlo method on multiple object tracking is not feasible because it does not define an obvious way to identify individual hypotheses.

In our previous work [17], we used different colored fingertip markers for tracking. Using colored markers, it is easy to use the standard particle filter to track each marker separately (because of the different colors). However, in the case of markerless tracking, particles are not distributed to each fingertip consistently since each fingertip represents the same hypothesis. To solve this problem, we extend the standard particle filter by applying a deterministic clustering approach as proposed in [7]. We create rectangular ROIs in each fingertip, and then we distribute the particles only inside the corresponding ROIs (while the standard particle filer will distribute particles all over the image).

3.3.1 Clustering algorithm

As explained in [8], the idea of clustering is to create rectangular ROIs by determining if the contours found in the fingertip probability map, i.e., $R_{\text{normalized}}(x,y)$ are consistent enough using a buffer. The intensity in the gray scale image illustrates the probabilities of the fingertips (higher brightness means higher probability). In this way, after we compute the gray scale image of the fingertip probability map, contours are extracted from the *FindContours* function implemented in the Intel OpenCV library. In other words, contours meant the area of high probability of the fingertips. Every contour found is stored in the following vector: $Y_t = \left\{y_{1,t}^T, \ldots, y_{m_t,t}^T, \ldots, y_{M_t,t}^T\right\}^T$. We called this $Y_t = \left\{y_{1,t}^T, \ldots, y_{m_t,t}^T, \ldots, y_{M_t,t}^T\right\}^T$ a contour vector. At any particular time, there are a total of M_t contours found from the fingertip probability map image. The system receives a contour vector $Y_t = \left\{y_{1,t}^T, \ldots, y_{m_t,t}^T, \ldots, y_{M_t,t}^T\right\}^T$ from the fingertip probability map (because the received contours may be noise also).

Denote a set of selected ROIs by $Z_t = \{Z_t^{(j)}, j = 1, \ldots, J_t\}$, where J_t is the number of regions we found at t within Y_t. Every region $Z_t^{(j)}$ is built according to a cluster of measurements obtained in Y_t and is stored in terms of a set of time and contour indices, i.e., pairs of indices (t,m_t). The concept is to group a collection of contours Y_t that are in the spatial vicinity of each other at various time steps. If the targets are divided obviously, the contours from their targets are clustered in the locations where the targets have been visited from $t - \tau$ to t. In this case, τ represents the buffer's width.

Given a set of independent contours Y_t, we need to find a set of selected regions. Denote a set of selected ROIs by $Z_t = \{Z_t^{(j)}, j = 1, \ldots, J_t\}$, where J_t is the number of ROIs found at t within Y_t. The jth region $Z_t^{(j)}$ comprises $P_t^{(j)}$ contours at successive scans in Y_t that are possible to obtain from the true interesting targets. The concept is to put a collection of contours Y_t together. Again, if the targets are divided clearly, the contours from their targets are clustered in places where the targets have been potentially visited from $t - \tau$ to t, where τ is the width of the buffer.

Next, we build each region $Z_t^{(j)}$ according to a cluster of contours received in Y_t. It is then stored in terms of a set of time and contour indices, i.e., pairs of indices (t,m_t). We denote the mth contour of $y_{t'+1}$ and the lth contour of $y_{t'}$ by $y_{m,t'+1}$ and $y_{l,t'+1}$, respectively. The normalized distance $d_{m,l}(t'+1, t')$ between $y_{m,t'+1}$ and $y_{l,t'+1}$ can be calculated from the intersection area between two contours. Our assumption is if the intersection area of two contours is high enough, these two contours should be grouped into the same cluster $Z_t^{(j)}$ (so the normalized distance $d_{m,l}(t'+1, t')$ will be set low). The minimum distance between two contours is also determined to calculate the normalized distance $d_{m,l}(t'+1, t')$. For every contour of $y_{t'+1}$, a set of normalized distances $\left\{d_{m,l}(t'+1,t'), t'\right\}_{m=1}^{M_{t'+1}}$ is obtained, where $m \in \{1, \ldots, M_{t'+1}\}$. It is important to note that $d_{m^*,l}(t'+1, t')$, $m^* \in \{1, \ldots, M_{t'+1}\}$ is the set minimum. The contours $\{y_{m^*,t'+1}, y_{l,t'+1}\}$ and contour indices $(t'+1, m^*_{t'+1})$ and $(t', l_{t'})$ are clustered together according to

$$0 \leq d_{m*,l}(t'+1, t') \leq \eta_0 \quad (5)$$

where η_0 represents a given threshold.

After we detect the ROIs, their classification is performed. We classify them differently if they are noise or ROIs. The ROIs we mentioned are possibly both active and inactive. Thus, we carefully determine this issue also. In order to decide this, we find a relationship

between the active tracks and the regions that we are interested in. By continuously finding the association and determining the appearance and disappearance of the regions, the system can recognize the number of tracking targets (this case is tips of the fingers) for different stages. Figure 2 depicts an example of gesture hand and fingertip recognition using the deterministic clustering algorithm.

3.3.2 Sequential Monte Carlo

In fact, there are two possible ways to use skin color probability in the particle filter step. Firstly, we can use the skin color probability itself. Secondly, we do threshold before and then use the binarized image. However, in our implementation, we use the second way in this paper. Each sample is propagated from the set s'_{t-1} according to

$$s_t^{(n)} = g\left(s_t'^{(n)}\right) + E \tag{6}$$

where E is Gaussian noise and $g(s'^{(n)}_t)$ is a propagation function. We use the noise information as the propagation function, i.e., $g(x) = x$. Figure 3 presents an example of finger tracking using the extended particle filter. After that, weights are generated by using the probabilities of fingertips from Equation 4. $p(X_t)$ represents the probability density function. Then, the sample set representation $\{(s_t^{(n)}, \pi_t^{(n)})\}$ of the state density for time t is calculated according to

$$\pi_t^{(n)} = p\left(X_t = s_t^{(n)}\right) = R_{\text{normalized}}(x, y) \tag{7}$$

where $p(X_t = s_t^{(n)})$ represents the probability that a fingertip is at position $s_t^{(n)}$. After that, just similarly as a normal particle filter process, the total weights are normalized. Moments of the pixel recognized are calculated at time-step t according to

$$\varepsilon[f(X_t)] = \Sigma_{n=1}^{N} \pi_t^{(n)} s_t^{(n)} \tag{8}$$

where N is sample that has been built and $\varepsilon[f(X_t)]$ is the tip of finger's centroid. Using the aforementioned framework, it enables us to track and achieve recognition.

4 Results

Figure 4 shows an example tracking of such online experimental results from the total 300 frames. The reported experimental result was run online using an Intel® Core™ i5-3317U Processor at 1.70 GHz. The top-left image represents the input images. This input image is captured from a camera. Note that the camera we used has 320×240 display resolution. We capture a scene where a user is showing his hand in front of the camera. The top-right image represents the hand segmentation adaptively. The bottom-left image shows the fingertip probability map. The intensity in this gray scale image represents the probabilities of the tips of the fingers. Higher brightness is higher probability, while lower brightness is lower probability. After performing the clustering algorithm and extended particle filter, the tracked results of fingertips are finally shown in the bottom-right images. The number of particles in the system is 300 particles. From our experimental results, this number of particles is suitable for the proposed methodology.

First, processing time is an important aspect of many embedded systems, especially if we would like to apply the vision-based method to use in embedded systems. However, the computation time for the sequence shown is real time (approximately 12 frames per second without optimization). From this processing time, it is quite convenient to implement the proposed method to use in the embedded systems architecture, especially embedded robotic systems. This is because robots using embedded systems usually need an image processing-based algorithm that can run in real time, or nearly real time. Thus, our experimental speed indicates that the proposed method can support embedded systems positively in this aspect.

The second issue about embedded systems is power. Any system that requires too much electric power is not feasible for embedded robotic systems. In our system, we test to run the system with a portable laptop using an Intel® Core™ i5-3317U processor. The system does not need any additional power. In fact, the system can

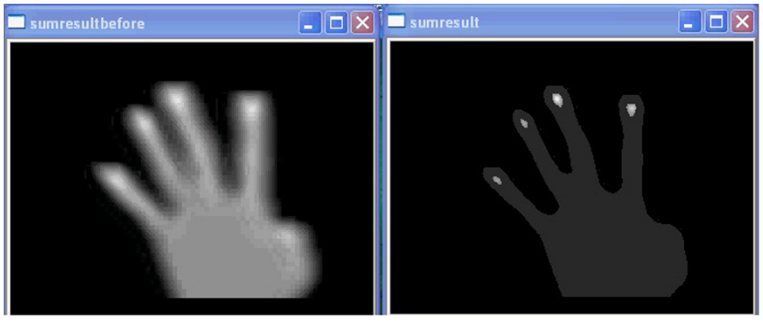

Figure 2 The deterministic clustering algorithm is used for gesture hand and fingertips recognition.

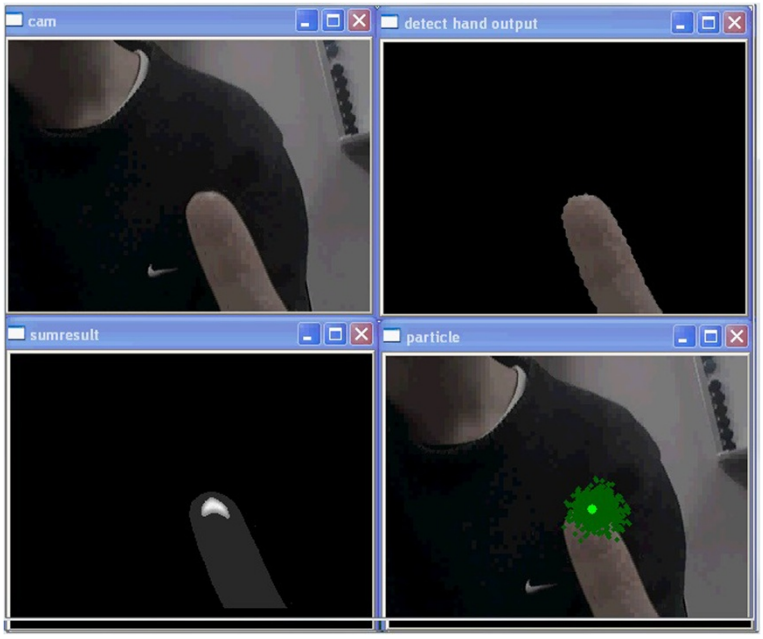

Figure 3 An extended particle filter is utilized for fingering tracking and recognition.

be powered portably from a lightweight tablet requiring only small amount of power. The laptop battery we used is eight-cell, 14.8 V 47 Wh/3,060 mAh lithium ion battery. It lasts at least 3 h when fully running the proposed tracking algorithm. Note that when it is not running the process, the battery lasts for approximately 4 h. For recharging the battery, the battery we used takes only 2 h which is also obviously convenient for utilizing in many smart embedded robots. Thus, from our experiments, the autonomy of the battery is practical for embedded applications even when it is fully running the tracking method. This means that the proposed vision-based method can easily apply to use in robots in terms of power for battery-powered embedded systems.

At the commencement of the experiment, a user enters the camera view field. Then he starts to change

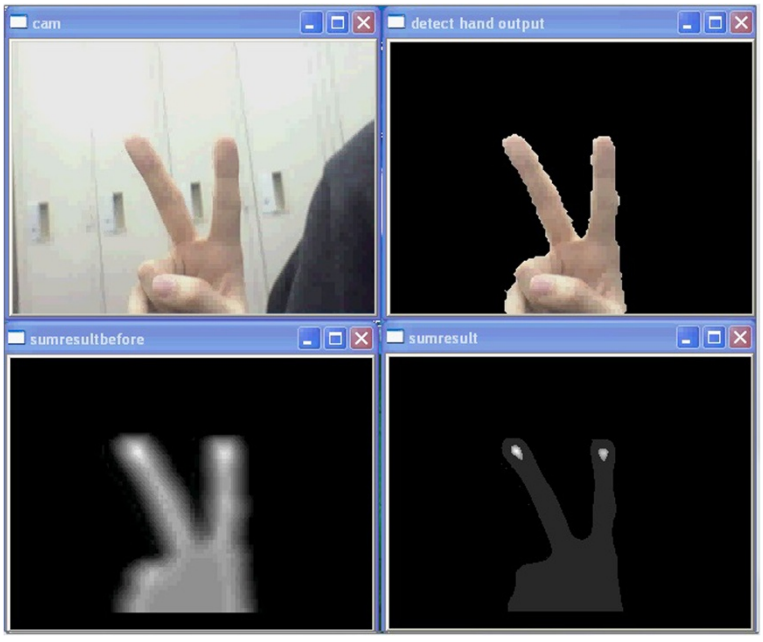

Figure 4 Representative snapshot at the commencement of the experiment while a user is showing his two fingers.

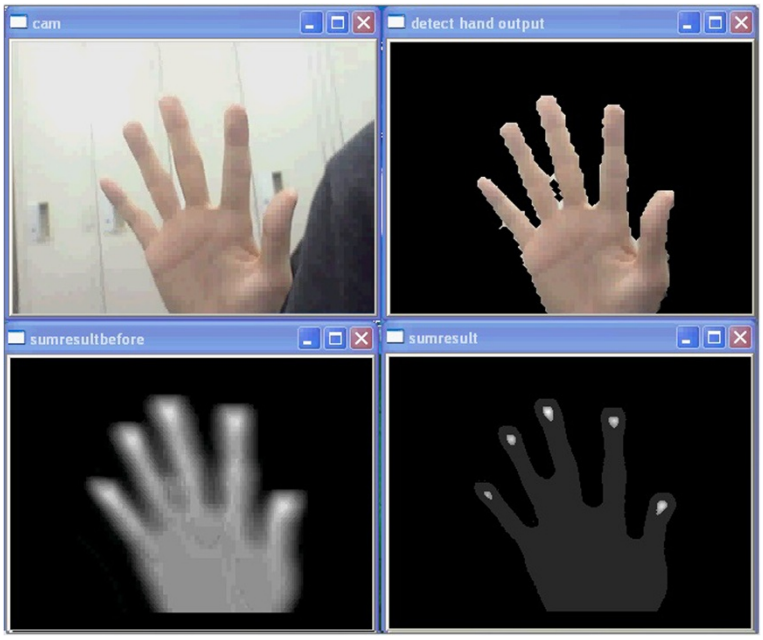

Figure 5 A user starts to change his hand to show five fingers clearly.

his hand in different poses. In our method, the number of detected ROIs can be varied according to the number of fingertips appeared in the input images (the number of ROIs is automatically found by the algorithm described in the previous section). For example, there are five fingertips appearing in Figure 5, while there are three and four fingertips appearing in Figures 6 and 7, respectively. However, the system is able to automatically determine the accurate number of appeared fingertips. The experiments have revealed that the system can successfully track the fingertip positions even when the luminance markedly changes from the off-line phase.

In order to evaluate quantitatively the accuracy of this method presented, we select 50 frames from 300 consecutive frames for evaluation, as depicted in Table 1. The predicted trajectory positions found by using the

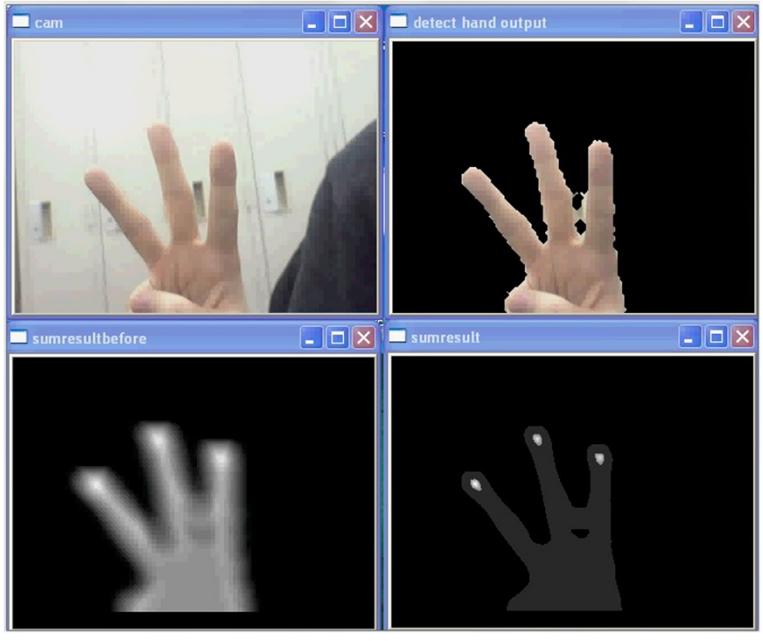

Figure 6 The number of detected ROIs can be varied, as correctly as the number of appeared fingers.

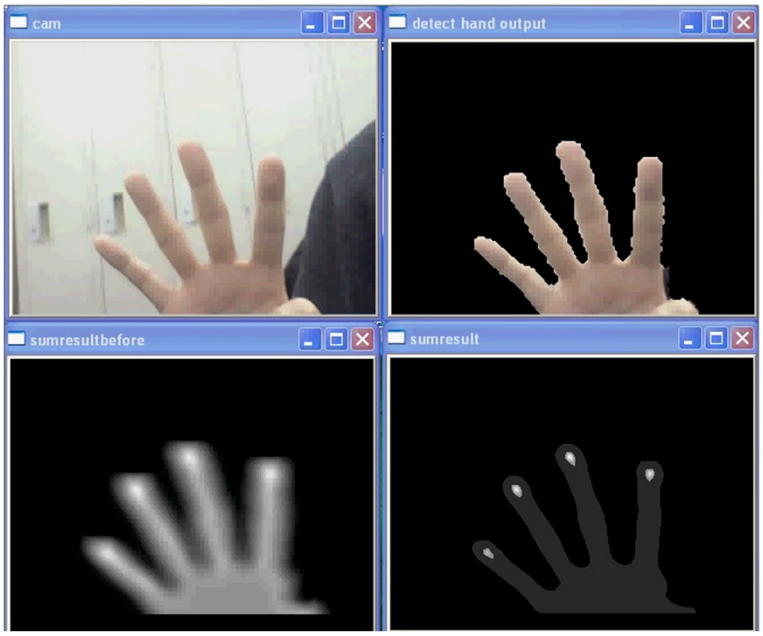

Figure 7 The tracker can automatically determine the correct number of appeared fingertips.

proposed tracking method are compared to the manually measured ground truth positions (actual). Such ground truth measurements are obtained by manually selecting the positions of the fingertips by mouse clicks. The positions of the tracked tips of the fingers are received by our system. Then we determine the Euclidean distance errors from 320×240 total image size in pixels. After obtaining the distance errors in each image, the mean distance error is computed. The standard derivation error is also calculated. It can be seen that the forefinger introduced the maximum mean error at 11.23 pixels, if comparing to the other fingers (5.31 pixels for the little finger, 7.42 pixels for the ring finger, 9.71 pixels for the middle finger, and 8.65 pixels for the thumb). This is because the forefinger usually moves quite fast in this experiment, comparing relatively to the movements of other fingers. So, it gives that the sequential Monte Carlo we used may not perform perfectly when the tracking objects move too quickly.

In this experiment, we use a unique input which is different from other inputs of the previous methods. Thus, it is not easy to compare directly the experimental results exactly to the results obtained by other algorithms. However, even though we do not compare directly to the same sequence of experimental input with other conventional methods, it is obviously seen that our proposed method outperforms qualitatively the previous methods, such as the results obtained by [6]. Also, although we do not use the same measurement with [3], with the numbers in Table 1, it is clear that our algorithm outperforms quantitatively the method presented in [3]. We believe these errors presented in Table 1 are sufficiently accurate to make the proposed framework a suitable methodology for human hand motion recognition and fingertip tracking.

5 Conclusions

This paper has developed an algorithm that tracks the positions of the hand and fingertips accurately. The skin-colored region of a user is segmented by applying a Bayesian classifier adaptively and automatically. After that, a matching algorithm is used to determine the probabilities of the fingertips based on their primitives. Following this, we extend the particle filter by using a deterministic clustering algorithm for tracking fingertips. The experimental results have shown that the proposed methodology is effective even with non-uniform backgrounds. The substantial analysis of the proposed method applied in embedded context, such as power, processing time, and the autonomy of battery-operated equipment, has also been

Table 1 Mean error and standard derivation in five fingertips

	Little finger	Ring finger	Middle finger	Fore finger	Thumb
Mean error (pixels)	5.31	7.42	9.71	11.23	8.65
Standard deviation	3.12	5.41	3.34	7.53	6.22

discussed. We believe that the proposed system can reach acceptably accurate results. However, we plan to solve the finger self-occlusion while using multi-cameras. This usually happens when using more than two cameras for stereo images. As part of our future work, we also intend to use this implementation to further develop the associated virtual-reality applications and related embedded robotic systems such as in [18] and [19].

Competing interests
The author declares that he has no competing interests.

Received: 12 March 2014 Accepted: 13 May 2014
Published: 3 July 2014

References

1. K Oka, Y Sato, Real-time modeling of face deformation for 3D head pose estimation, in *Proceedings of the International Workshop on Analysis and Modelling of Faces and Gestures, AMFG '05, Beijing, China, 16 October 2005* (Springer, Berlin Heidelberg, 2005), pp. 308–320
2. J Letessier, F Bérard, Visual tracking of bare fingers for interactive surfaces, in *Proceedings of the ACM Symposium on User Interface Software and Technology, ACM/UIST '04, Santa Fe, NM*, 2004, pp. 119–122. ISBN 1-58113-957-8
3. J Mackie, B McCane, Finger detection with decision trees, in *Proceedings of the Image and Vision Computing New Zealand, IVCNZ '04, Akaroa, New Zealand*, 2004, pp. 399–403
4. C Kerdvibulvech, Real-time framework of hand tracking based on distance transform, in *Proceedings of the 11th International Conference on Pattern Recognition and Image Analysis, PRIA-11 '13, Samara, Russian Federation, 23–28 September 2013* (RAS and Springer, Pleiades, 2013), pp. 590–593
5. KE Papoutsakis, AA Argyros, Integrating tracking with fine object segmentation. Image. Vision Comput. **31**(10), 771–785 (2013)
6. CM Baris, N da Vitoria Lobo, Open hand detection in a cluttered single image using finger primitives, in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition Workshop, CVPR '06 Workshop, New York, NY*, 2006, p. 148. ISBN 0-7695-2646-2
7. TCT Chan, H-C So, KC Hom, Particle filtering based approach for landmine detection using ground penetrating radar. IEEE Trans. Geosci. Remote Sens. **46**(11), 3739–3755 (2008)
8. C Kerdvibulvech, H Saito, Model-based hand tracking by chamfer distance and adaptive color learning using particle filter. EURASIP J. Image Video Process (2009). (Springer, Hindawi, New York, 2009). Article ID 724947, 10 pages
9. AM Martínez, RB Wilbur, R Shay, AC Kak, Purdue RVL-SLLL ASL database for automatic recognition of American sign language, in *Proceedings of the IEEE International Conference on Multimodal Interfaces, Pittsburgh, PA*, 2002, pp. 167–172
10. M Matti, H Jari, F Li-Xin, Fan finger tracking for gestural interaction in mobile devices, in *18th Scandinavian Conference on Image Analysis, SCIA '13, Espoo, Finland, 17–20 June 2013*. Lecture Notes in Computer Science, vol. 7944 (Springer, Heidelberg, 2013), pp. 329–338
11. P Krejov, R Bowden, Multitouchless: real-time fingertip detection and tracking using geodesic maxima, in *Proceedings of the 10th IEEE International Conference on Automatic Face and Gesture Recognition, FG '13, Shanghai, China*, 2013, pp. 1–7
12. C Kereliuk, B Scherrer, V Verfaille, P Depalle, MM Wanderley, Indirect acquisition of fingerings of harmonic notes on the flute, in *Proceedings of the International Computer Music Conference, ICMC '07, Copenhagen, Denmark*, vol. I, 2007, pp. 263–266
13. C Kerdvibulvech, H Saito, Markerless guitarist fingertip detection using a Bayesian classifier and a template matching for supporting guitarists, in *Proceedings of the 10th ACM/IEEE Virtual Reality International Conference, VRIC '08, Laval, France*, 2008, pp. 201–208
14. A Asthana, TK Marks, MJ Jones, KH Tieu, MV Rohith, Fully automatic pose-invariant face recognition via 3D pose normalization, in *Proceedings of the IEEE International Conference on Computer Vision, ICCV '11, Barcelona, Spain*, 2011, pp. 937–944
15. X Wei, SL Phung, A Bouzerdoum, Object segmentation and classification using 3-D range camera. J. Vis. Commun. Image Represent. **25**(1), 74–85 (2014)
16. S Shi, L Wang, J W-q, Y Zhao, Yuanmeng color night vision based on color transfer in YUV color space, in *Proceedings of the International Symposium on Photoelectronic Detection and Imaging, Beijing, China, 9 September 2007*, vol. 6623, 2007, p. 66230B. doi:10.1117/12.791275
17. C Kerdvibulvech, H Saito, Vision-based guitarist fingering tracking using a Bayesian classifier and particle filters, in *IEEE Pacific-Rim Symposium on Image and Video Technology, PSIVT '07, Santiago, Chile, 17–19 December 2007*. Lecture Notes in Computer Science, vol. 4872 (Springer, Berlin Heidelberg, 2007), pp. 625–638. ISBN 978-3-540-77128-9
18. M Bianchi, P Salaris, A Bicchi, Synergy-based hand pose sensing: optimal glove design. Int. J. Robot. Res. **32**(4), 396–406 (2013)
19. J Huang, A Raabe, K Huang, C Buckl, A Knoll, *A framework for reliability-aware design exploration for MPSOC based systems*. Design Automation for Embedded Systems (DAEM), vol. 16, no. 4 (Springer Science+Business Media, New York, 2013), pp. 189–220. doi:10.1007/s10617-013-9105-6

doi:10.1186/s13639-014-0018-7
Cite this article as: Kerdvibulvech: **A methodology for hand and finger motion analysis using adaptive probabilistic models.** *EURASIP Journal on Embedded Systems* 2014 **2014**:18.