Multimed Tools Appl (2016) 75:10059–10076
DOI 10.1007/s11042-015-3061-z

Vision based multi-user human computer interaction

Shyh-Kuang Ueng[1] · **Guan-Zhi Chen**[1]

Received: 27 May 2015 / Revised: 9 October 2015 / Accepted: 3 November 2015 /
Published online: 9 November 2015

Abstract This paper proposes a vision-based *Multi-user Human Computer Interaction* (HCI) method for creating augmented reality user interfaces. In the HCI session, one of the users' hands is selected as the *active* hand. The fingers of the active hand are employed as input devices to trigger functionalities of the application program. To share the token of interaction among the users, the HCI session is modeled as a Finite State Machine (FSM). The FSM is composed of the *initial* and *steady* states. In the initial state, the FSM identifies the active hand by tracking the hand with the maximum moving speed. Then the FSM enters the steady state to carry out the HCI session. At the end of each individual HCI cycle, the FSM polls requests from other hands for acquiring the role of active hand. If such requests are sensed, the FSM returns to the initial state to track a new active hand. Otherwise, the HCI session is continuously carried out by the current active hand. Test results show that the resultant user interface is efficient, flexible and practical for users with problems on using ordinary input devices. In a desk-top computer equipped with a 640 × 480 resolution web-camera, the HCI session can be successfully conducted when the operation distance ranges from 30 to 90 cm.

Keywords Human computer interaction · Hand tracking · Finger detection · Multi-user · Healthcare systems

1 Introduction

Using hand gestures and fingertips as input devices is a natural and convenient way in Human Computer Interaction (HCI). In [13], Manresa et al. proposed a hand tracking and gesture recognition method for the control of a video game. In the work of [10], hand gestures were used to create a coordinate system in a mark-less Augmented Reality (AR) system and to control the motions of virtual objects. In [1], Azuma surveyed the usage of hand gestures as

✉ Shyh-Kuang Ueng
skueng@mail.ntou.edu.tw

[1] Department of Computer Science & Engineering, National Taiwan Ocean University, Keelung City 202, Taiwan

input devices in AR applications. Numerous other applications of hand gesture recognition and fingertip tracking were reviewed in the papers of [2, 4].

Capturing hand gestures and fingertips in real time can be accomplished by using capacitive sensors [15, 16], data gloves or Computer Vision (CV) based methods [2–4, 12]. CV based methods are usually more convenient, because they are unrestricted and less expensive. They can be implemented in desktop machines, notebook computers, smart phones, and other wearable computational devices. However, in a complex and dynamic scene, where users' hands, faces and body parts appear and move constantly, CV based methods may produce inaccurate results. Thus, some CV based methods are only capable of supporting single hand interaction in stable and simple scenes. Even so, the variation of hand orientation and gesture may still cause difficulties for hand-tracking and finger-detection. To subdue these problems, we can limit the number of comprehensible hand gestures and assume that the system is operated in a monotonic environment where only one hand shows up. But the resultant HCI system is less user-friendly and its usability is greatly reduced.

In this paper, we present a CV based HCI method to create Augmented Reality (AR) interfaces to help people who cannot properly use traditional input devices to interact with computers. In the working environment, multiple users can simultaneously participate in the HCI session. The users use their fingertips to press the buttons of an AR interface to trigger functionalities of the application program. They can alternatively interact with the system or complete a procedure together. To realize the HCI environment, we have to resolve two problems: First, the images captured by the camera are very complex because of the human hands, faces and body parts appearing in the scene. Segmenting skin regions, tracking hands and detecting fingertips in real time become challenging and cannot be easily conquered by using conventional CV based hand tracking methods. The second problem is to share the authority of interaction with the computer among the users so that everyone has chances to operate the system. To overcome the first problem, we employ a skin color model and develop a *redness test* to segment skin regions. Then the hand-tracking process is carried out by using an *optical flow* method, instead of template based algorithms. Hence pattern matching for identifying hand gestures is unnecessary. Subsequently, we explore the geometrical features of fingers and utilize them to simplify the finger detection process so that the computation can be accomplished in real time. To overcome the second problem, only one of the users' hands is allowed to interact with the computer in each HCI cycle. This hand is designated as the *active hand*. Fingers of the active hand are treated as input devices to trigger functionalities of the application program. However, the role of active hand can be transferred at the end of each HCI cycle so that users can alternatively interact with the computer.

In our implementation, the multi-user HCI environment is modeled as a Finite State Machine (FSM). The FSM is composed of two states, the *initial* and *steady* states. In the initial state, the FSM fetches two consecutive frames from the camera at first. Then it segments the skin regions and constructs the contours of the skin regions. At the following step, an optical flow method is employed to calculate the speeds of these skin region contours. The contour with the highest speed is regarded as the contour of the active hand. Once the active hand is located, its center position, contour, speed and bounding box are computed and recorded. After completing the hand-tracking process, the FSM enters the steady state to conduct the HCI session. In each HCI cycle, the FSM reads a video frame at first. The speed and center position of the current active hand are used to create a search window in the scene. The tracking of the active hand is confined within the search window so that the computation is sped-up and the current active hand has a higher priority to keep its role. Finally, the

fingertips of the active hand are detected and used as input devices to trigger functionalities of the application program. As the current HCI cycle is completed, a new frame is fetched and the hand tracking, fingertip detection and HCI are repeated. The FSM will leave the steady state if any of the following events occurs: (1) The active hand is lost in the incoming frame. (2) Other hands want to win the role of active hand and the FSM has sensed the intentions. The FSM responses to these events by switching back to the initial state to track a new active hand so that another user can win the authority to interact with the computer. The brief transition diagram of the FSM is shown in Fig. 1.

The essential techniques involved in building the multi-user HCI environment include: the skin region segmentation, moving speed computation for skin regions, fingertip detection, and active hand transferring. These algorithms are to be described in Section 2. To verify the usability of the proposed multi-hand HCI mechanism, a home-based voice health caring system is built by using the technology so that people with motion problems can operate the application program by using their hands and fingertips. The voice health caring system and its multi-user HCI interface are presented in Section 3. Other testing results and analysis are also shown in Section 3 to reveal the usability and limitation of the proposed HCI method. The tests are conducted to convey the rationale of the redness test and to compute the feasible operational distance of the proposed system. Beside these materials, comparison of the proposed method with four commercial products supporting HCI is contained in Section 3 too. The conclusion and future work of multi-user HCI are presented in Section 4.

2 Material and methods

2.1 Initial state hand tracking

The goals of the initial state hand-tracking are to locate the active hand and to compute its contour, center position and moving speed. The flowchart of the initial state hand tracking is shown in Fig. 2. At first, two RGB images are fetched from the camera. Then the skin regions are segmented. In the following step, the contours of all skin regions are calculated and an optical flow method is adopted to compute the moving speeds of these contours. The contour with the maximum moving speed is regarded as the contour of the active hand.

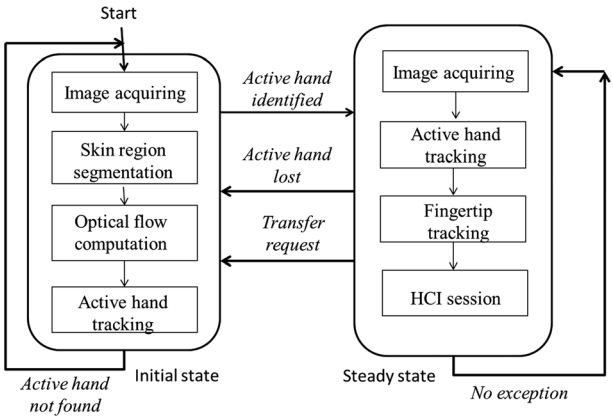


Fig. 1 The multi-user HCI is modeled as a 2-state FSM

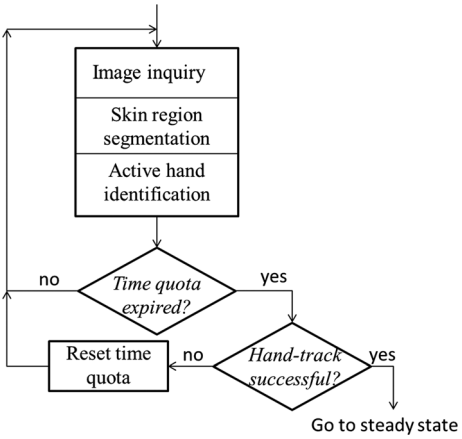


Fig. 2 Flowchart of the initial state hand tracking

2.1.1 Skin region segmentation

The first task to locate the active hand is to segment the skin regions. The input image is stored in both the RGB and YCbCr formats. The resulting images are denoted as $I_{RGB}(t)$ and $I_{YCbCr}(t)$, where t represents the time step. At first, the Gaussian skin color model proposed in [19] is employed to extract potential skin pixels in $I_{YCbCr}(t)$. We use the Cb and Cr components to compute the probability of being a skin pixel for each pixel. If the probability is higher than a predefined threshold (0.86 in our implementation), the pixel is registered as a potential skin pixel.

However, the skin color model of [19] does not perfectly match the distribution of human skin colors. Reddish pixels in the background may be falsely classified as skin pixels. Thus we augment the skin pixel extraction process with a *redness test* to reduce such errors. The redness test is carried out as follows: If a pixel is classified as a potential skin pixel, its RGB components are retrieved from $I_{RGB}(t)$ and substituted into the following test:

$$\left(\frac{R}{G}\geq q\right) and \left(\frac{R}{B}\geq q\right)? \tag{1}$$

where R, G and B are the RGB components of the pixel. The threshold q is used to measure the dominance of the R component. Based on our experiments, the value of q is set as 1.5 to produce the best results. If the redness test is true, the intensity of the R component exceeds the intensities of the G and B components by more than 50 %. It implies that this pixel is a reddish pixel but not a human skin pixel and is excluded from the skin regions. Otherwise it is classified as a skin pixel. Several tests had been conducted to measure the threshold q of the redness test. Analysis and discussion will be presented in Section 3.3.

The skin pixel segmentation process produces a binary image, $I_{bin}(t)$, where the 1-pixels stand for skin pixels. This binary image is then under an erosion operation by using a 3 × 3 *4-connected* structure to delete small skin regions and thin edges. At the following step, a dilation process is carried out to fill small holes and connect broken edges using the same connected structure. Finally, the contours of the remaining skin regions are computed and recorded in another binary image, $I_{cont}(t)$, in which the 1-pixels represent the boundaries of the skin regions. An example of the proposed skin region segmentation is shown in Fig. 3. The

(a) Raw image (b) Skin region (c) Active hand

Fig. 3 Skin region extraction and hand tracking, (**a**) the raw image, (**b**) skin regions, (rendered in *white color*), (**c**) the identified active hand (enclosed by a *red bounding box*)

raw image is displayed in part (a), and the extracted skin regions are super-imposed on the raw image and shown in part (b). The skin regions are rendered in white color. As the image shows, the skin regions include hands, faces and body parts.

2.1.2 Active hand tracking using an optical flow method

After the skin regions have been extracted, the FSM starts to identify the active hand. As shown in Fig. 3, the skin regions include not only the users' hands but also their faces and body parts. Some researchers use template matching methods to locate hands. However, the users' hands may be in various orientations and gestures. It is hard to create a complete database of hand templates. Even if such a database is available, the matching would be too computational intensive. To speed up the process, we adopt the following two principles in tracking the active hand: (1) *Users' hands usually move faster than their faces and body parts.* (2) *Users try to gain the authority of interaction with the computer by raising their hands, waving their hands, or changing hand gestures.* Based on these principles, the skin region with maximum moving speed is regarded as the active hand.

To locate the active hand, the previous and current RGB images $I_{RGB}(t)$ and $I_{RGB}(t-1)$ are converted into grey-level images. Then an optical flow computation is conducted to calculate the velocity of each pixel using these two grey-level images. The speeds of the pixels are kept in a 2D array $I_{speed}(t)$. At the following step, $I_{speed}(t)$ is modulated with the contour image $I_{cont}(t)$ to obtain the speeds of the contour pixels. The speed of a skin region is set to the average speed of its contour pixels. Finally, the skin region with the maximum speed is regarded as the active hand.

An example of tracking active hand is illustrated in Fig. 3. The raw image and the skin regions are displayed in parts (a) and (b). The user in the right side waves his right hand while the user in the left side keeps his hands motionless. Thus the right hand of the user in the right side has the maximum moving speed among the skin regions and is designated as the active hand. The tracked active hand is shown in part (c). The red rectangle is the bounding box of the active hand.

2.1.3 Iterative hand tracking

In some occasions, the tracking of active hand may fail. For examples, no skin region is present in the scene or the speeds of all the skin regions are zero. To overcome these problems, the initial state hand tracking is conducted in an *iterative* manner as shown in the flowchart of Fig. 2. Before the hand tracking starts, a time quota is assigned to the process. The hand

tracking is repeatedly performed until the time quota has expired. If multiple active hands have been detected, the lastest-found active hand is regarded as the active hand. If no active hand is found and the time quota has expired, the time quota is reset and the process is restarted. The time quota should be selected with care. If it is too small, the system may fail to find any active hand. On the other hand, if the time quota is too large, users will feel the latency. Based on our experiments, we set the time quota to 0.3 second which is equivalent to the time of acquiring 10 frames from the camera.

2.2 Steady state hand tracking

Once the active hand has been identified, the FSM enters the steady state to conduct the HCI session. The flowchart of the steady state hand tracking is shown in Fig. 4. At first, a search window is constructed for tracking the active hand. Once the active hand has been located, its fingertips are detected and used as the input devices to control the application program. At the end of each HCI cycle, the FSM fetches a new image and checks whether there are intentions from other hands to win the role of active hand. If no such request exists, the FSM resides in the steady state and repeats the HCI cycles. Otherwise the FSM migrates to the initial state to identify a new active hand.

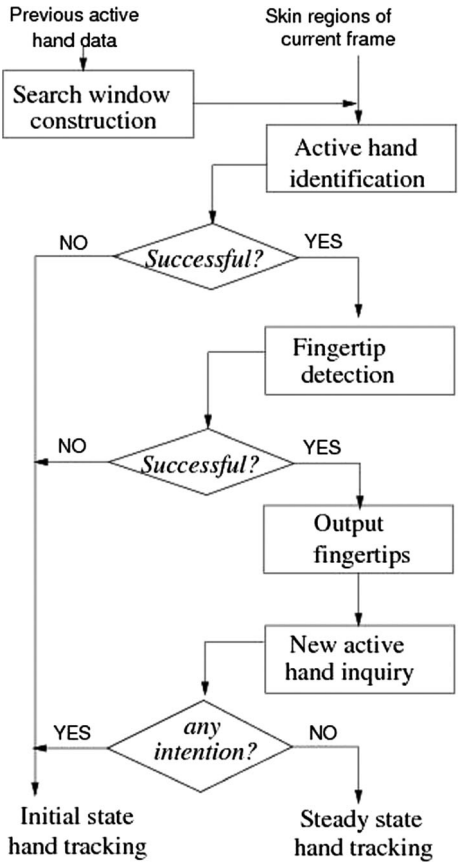


Fig. 4 The flowchart of steady state hand tracking

2.2.1 The search window of an active hand

We assume that the active hand does not move too much between two consecutive frames. It can be located within the neighborhood of its previous location. At first, we enlarge the bounding box of the active hand in the x and y directions by D pixels to construct a search window. The value of D is defined by:

$$D = \Delta t * v, \tag{2}$$

where Δt is the inter-frame time and v is the moving speed of the active hand, which is computed in the initial state hand tracking. Then we search skin regions inside this window. If exactly one skin region is detected, the skin region is regarded as the active hand's skin region. If multiple skin regions are detected, the skin region which is closest to the active hand position at the previous time step is regarded as the active hand. As the active hand is located, its geometrical center and bounding box are recorded and the finger detection process starts.

In the work of [10, 11, 20, 21], similar methods were proposed to locate hands in scenes. In their methods, the search window is created by using the velocities of hands. Thus the search window is translated in the direction of velocity in the new frame. In our HCI system, we assume that user may move his hand in any directions between two frames. Thus we use the speed and geometrical center of the active hand at the previous time step to create the search window to avoid tracking the active hand in wrong directions.

2.2.2 Loss of the active hand

On some occasions, exceptions may occur in the hand tracking process and the FSM has to interrupt the HCI session and perform the initial hand tracking again: (1) the active hand moves too fast so that its position is out of the search window, (2) the active hand overlaps users' body parts or faces and no fingertip can be detected. These cases are not handled by using elaborate methods in this work for the sake of quick response time. We let the user resolve these problems: As the FSM switches back to the initial state to track a new active hand, the user can wave his hand slightly to regain the control of interaction. An example is shown in Fig. 5 to illustrate the recovery of the active hand role. In part (a), the active hand overlaps the user's face and the fingertip detection procedure locates no fingertip. The FSM is enforced to restart the active hand tracking computation. At this moment, the user waves his hand and moves his hand away from his face so that his hand possesses the control again. The resulting image is displayed in part (b). Once the active hand has been tracked again, the FSM enters the steady state and the user can continue the HCI session. The results are as shown in part (c).

(a) Overlap of hand & face (b) Active hand tracking (c) Active hand recovered

Fig. 5 Loss of active hand caused by overlapping and the recovery process

2.2.3 Transfer of active hand

In a multi-user HCI session, the authority of interaction needs to be transferred among the users. For example, the current user is a learner and the second user is a teacher. They alternatively interact with the computer. In the proposed HCI method, a user can take over the authority of the HCI session from the current user by raising and waving one of his hands. Thus, if the skin regions in two consecutive frames vary significantly, an active hand transfer request is issued and the active hand has to be re-identified. We adopt a content-based algorithm to implement this active hand transferring mechanism. At the end of each HCI cycle, the FSM fetches a new image and extracts skin regions from it. Then the FSM computes the difference of the skin regions at the current and previous time steps :

$$I_{diff}(t) = I_{bin}(t) \oplus I_{bin}(t-1), \tag{3}$$

where $\oplus$ is the exclusive-or operator and $I_{bin}(t)$ and $I_{bin}(t-1)$ are the binary skin region images. Then the number of the 1-pixels in the resultant image is counted. If the number exceeds 40 % of the number of the pixels in the bounding box of the active hand, the FSM assumes that there is a new hand intruding the scene or another hand moves significantly. The intention of the action is to request the authority of interaction. Thus the FSM switches to the initial state to identify the new active hand. In case that other hands and body parts move slowly or keep motionless, the difference of skin region will not exceed the threshold, and the tracking of a new active hand is not going to be initiated.

An example is shown in Fig. 6 to demonstrate the active hand transferring process. In part (a), the user in the right side possesses the authority of interaction. One of his hands is the designated as the active hand which is enclosed in the green bounding box. Then the user lowers his active hand while the user in the left side raises and waves his left hand to request the authority of interaction. Since the image difference, computed by using the aforementioned method, exceeds the threshold, the FSM switches to the initial state to track the new active hand. The new active hand is found and enclosed by a red bounding box, as shown in part (b). Finally, the FSM enters the steady state again and the second user uses his active hand to interact with the system. A snapshot is shown in part (c). The new active hand is enclosed by a green bounding box.

2.3 Fingertip detection

The fingertips of the active hand are used as pointing devices in the HCI environment. We use two geometrical features to locate the fingertips. The procedure and related issues are presented in the following subsections.

(a) Normal HCI session (b) Request for control (c) New active hand

Fig. 6 Active hand transferring in the proposed multi-user HCI method

2.3.1 Geometrical features of fingertips

Assume the active hand contour is composed of a list of pixels enumerated in the *counter-clockwise* order. At pixel P_i, we create two edges pointing to pixels P_{i+k} and P_{i-k}. Since the width of a finger is much smaller than its length, the angle spanned by the two edges is small if P_i is at a fingertip. Secondly, if P_i is at a fingertip position, it should be near the center of the arc connecting P_{i-k}, P_i and P_{i+k}. The left and right parts of the arc should be nearly symmetric and the lengths of the two edges are approximately equal.

Four cases are shown in Fig. 7 to explain the classification of fingertip: In case (a), P_i (the red point) possesses both the features and it is a fingertip. In case (b), the angle spanned by the two edges is too large and P_i is not a fingertip. In case (c), the lengths of the two edges are obviously different and the angle spanned by the edges is large. Therefore P_i is not a fingertip. In case (d), the angle is greater than 180° and P_i is not a fingertip but a concave position. Some researchers use curvatures of hand contours to locate fingertips, for examples, [10, 11]. Compared with their methods, ours is easier to implement and still applicable even the active hand contour is not smooth.

2.3.2 Implementation issues of fingertip detection

In our implementation, the value of k is set to 8–10 pixels depending on the size of the bounding box of the active hand. The threshold angle is set to 15°. The values of k and the threshold angle are decided by experiments. The normal operational distance from the users' hands to the camera is between 30 cm to 90 cm. If the value of k is too small, many points in

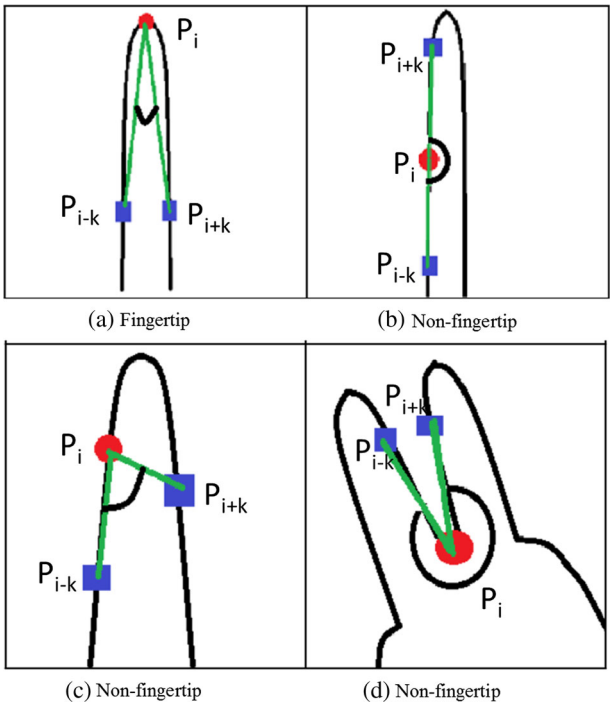


Fig. 7 Fingertip point classification, (**a**) a fingertip, (**b**) and (**c**) non-fingertips, (**d**) a concave point

the active hand contour will be falsely classified as fingertips. On the other hand, if k is too large, some fingertips will be missing in the detection results. To measure the angle spanned by the two edges, we compute the inner product of the directional vectors of the edges first. If the inner product is smaller than that of the threshold angle, P_i is not a fingertip point. If P_i passes the test, the cross-product of the directional vectors is computed. If the resulting vector points into the screen, P_i is a concave point and is ignored. Otherwise, the lengths of the edges are calculated. If the difference of the two edge lengths is less than 3 pixels, P_i is a fingertip position.

The active hand contour may contain many pixels. If the fingertip detection is performed at every pixel, the computation would be too time-consuming. To speed up the process, we detect fingertip for every 3 pixels. The pixels near a fingertip usually possess both the features. Thus several pixels may be identified as fingertip positions there. These pixels form a cluster. Based on this observation, we filter out isolated fingertip pixels to reduce noise and group the remaining fingertip pixels into a cluster. Then the geometrical center of the cluster is calculated and is regarded as the final result. Since the center of the point cluster is selected as the final fingertip, the detected fingertip will be lower than the real fingertip. An example of the fingertip detection process is shown in Fig. 8. In part (a), the detected fingertip points are shown. They form clusters around the fingertips. After the clustering process, the fingertips are located and shown in part (b). The fingertips are rendered as purple circles. Then the user changes the hand gesture and the fingertips are successfully detected in all the test cases. The results are displayed in parts (c) to (e).

If the arm shows up in the scene, the fingertip detection procedure may regard the intersection points of the arm contour and the window boundary as fingertips, as shown in Fig. 8. Our remedy is to ignore all fingertips found in the window boundary. Another method is to cut the arm by using the procedure proposed in [14]. For the sake of saving execution time, we have not implemented this function yet.

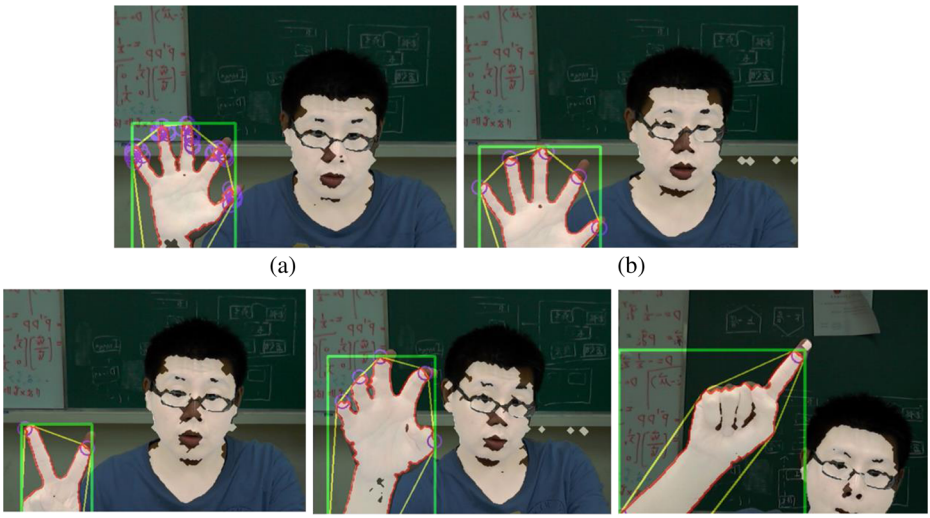


Fig. 8 Fingertip detection: (**a**) before clustering, (**b**) after clustering, (**c**)–(**e**) detecting fingertips in various hand gestures

3 Results and analysis

3.1 Application of the HCI method

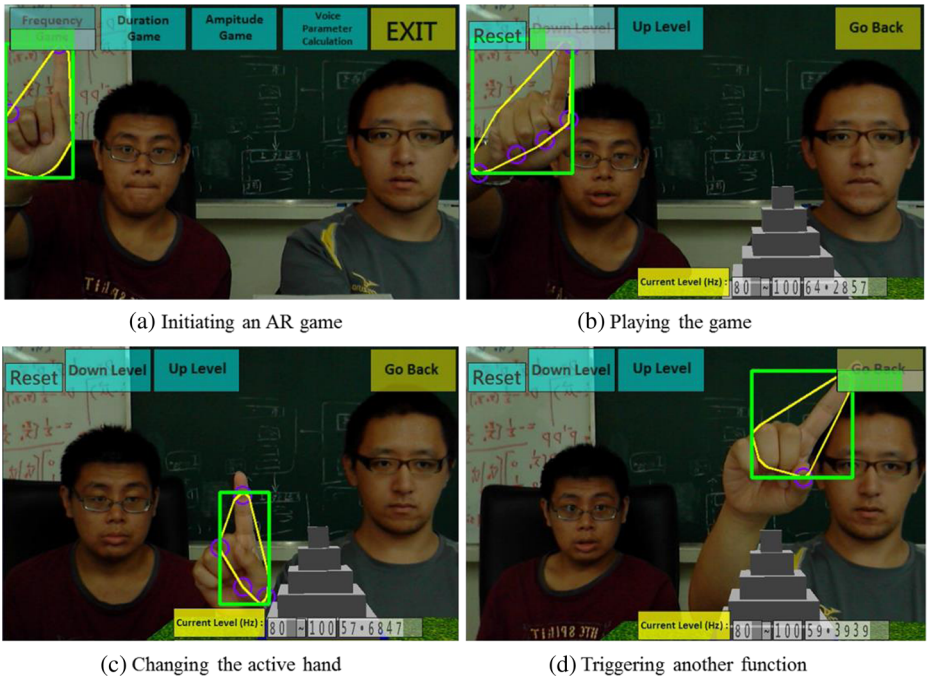
In the modern society, many people suffer from pathologic voice problems. Some of these problems can be cured by surgeries, medicines or physical therapies. After these treatments, the patients usually have to go through a series of voice diagnosis and uttering practice. Currently, there are several commercial systems focusing on this voice health caring procedure, for example the MDVP system [14]. These systems are expensive and have to be operated in very quiet environments. Furthermore their user interfaces are very complicated. Only experts and well trained physicians can operate these systems. As a result, these systems are available only in major medical centers, and patients can get treatments at most once a week. To improve the situation, we create a voice assessment system which is simpler and easier to use. The system can be operated in a living room with moderate noises so that the user can practice uttering and measure their voice qualities at home. The system fetches the user's voice from a microphone and computes voice parameters to measure the health conditions of the input voice. The results are shown by using visualization methods so that ordinary users can understand the assessment results. The system also offers various games for users to practice uttering. Therefore, users can gradually improve their voice qualities. The detail description of the voice health caring system can be found in [17].

After the system has been installed and tested in a lab, we found that some patients who suffer from strokes or neurological diseases may have difficulties to use the keyboard and mouse to interact with the system. To make the system more user-friendly, we utilize the proposed HCI method to create an AR user interface for the voice health caring system. In the interface, the users use their fingers to touch virtual buttons, which are drawn on the screen, to select menus, trigger functionalities, and control the computational flow. Some snapshots of operating the system are shown in Fig. 9. In part (a), the first user uses one of his fingers to trigger a training game. The scene of the game is displayed in part (b). At the same time, he selects a level of pitch to practice uttering. Then the second user requests the authority of interaction and assists the first user to select another game, as illustrated in part (c). From now on, the user interface is changed to a Chinese version, which is more comprehensible to the first user. The snapshot is displayed in the part (d).

The voice caring system and the multi-hand HCI interface are implemented on a desktop machine which is equipped with an AMD Phenom II 6 core 3.0 GHz CPU, a microphone and a Logitech HD Webcam C310 camera. The frame rate of the camera is 30 fps. The resolution of the input image is 640 × 480 and the AR scene and the user interface are rendered in an 800 × 800 window. Since the steady state hand tracking and the fingertip detection methods are very efficient, the operation is carried out in real time. The time required to track the active hand and detect fingertips in the steady state is less than 1/36 second. The most expensive computation is to calculate the voice parameters for the users. To avoid the calculation blocking the operation, the system is built up by using a multiple thread strategy.

3.2 The influence of the operational distance

Since the proposed HCI environment is CV based, its usablity is influenced by the distance between the users' hands and the camera. Before installing the system in laboratories, we have to figure out the best operational distances. In an experiment, we conducted a sequence of tests to calculate the feasible range of operational distance. In the experiment, three volunteers were



(a) Initiating an AR game (b) Playing the game

(c) Changing the active hand (d) Triggering another function

Fig. 9 An AR user interface using the proposed HCI method, (**a**) operated by the 1st user, (**b**) playing a game, (**c**) the 2nd user acquires the control, and (**d**) operated by the 2nd user

invited to run the voice health caring progam. The distance between the camera and the volunteers' hands were varied from 20 cm to 100 cm. In the tests, the users' hands were in two gestures: In the first gesture, the fingers were straight. In the second gesture, the fingers bended forward by about 60°. The hand tracking and finger detection processes were performmed for about 10 min for each test. Thus more than 300 times of initial state and steady state hand tracking and finger detection were performmed in one test case. The users moved hands in a vertical plane which was at the specified disatnce from the camera so that the operational distance was nearly the same. Numerical counters were added into the FSM to record the numbers of hand-tracking, successful handtracking and fingertip detections, and failed hand tracking and finger detections. The true positive rates of the active hand tracking and finger detection at different distances are shown in Table 1. For comparison the dection rates at various distances by using Leap Motion [7] are presented in Table 2. The results show that Leap Motion enjoys high detection rates when the operational distance is short. Its perforamce sharply falls if the operational distance is beyond its working range.

Since the hand-tracking is carried out by using the optical flow method and the skin color model, the active hand can always be identified if it moves faster than body parts and faces. On the other hand, finger detection rate is significantly reduced if the operation distance is too small or too large. An example is shown in part (a) of Fig. 10. In this case, one of the fingers is regarded as the active hand but not a fingertip. The reasons can be explained as follows: The widths of fingers in the images change as the operational distance is varied. Nonetheless, the edge length, k, used for detecting fingertips is between 8 and 10 pixels, as described in Section 2.3.2. The geometrical feature of fingertip, depicted in Fig. 7, is deteriorated if the operational distance is not within the ideal range. Though we can adaptively adjust the edge

Table 1 Hand-tracking & finger-detection rates using the proposed system

Process \ Distances (cm)	20	25	30	60	90	100
Finger detection (straight)	10 %	100 %	100 %	100 %	100 %	40 %
Finger detection (curled)	40 %	100 %	100 %	100 %	100 %	70 %
Hand tracking	100 %	100 %	100 %	100 %	100 %	100 %

length according to the operation distance. However, to sense the operational distance, extra hardwares are needed. It will increase the cost of the health caring system. Thus, this function is not implemented in our system yet. Instead, we conducted these experiments and concluded that the normal operational distance should be between 30 and 90 cm, and the optimal edge length k for fingertip detection is from 8 to 10 pixels within this range of operation distance.

One interesting point is that the finger detection rate of the curled hand is higher than the finger detection rate of the straight hand though the fingertip positions are slightly lower. It seems that the fingers of the curled hand possess higher curvatures so that geometrical features of fingertips are easier to be detected. As the hand moves away from the camera by more than 100 cm, the fingers become too thin in the images. The contour of the hand may be broken so that finger detection is less successful. But the curled hand still produces more fingertip features. Thus the finger detection rate of the curled hand is acceptable. This test result ensure that users can operate the health caring system even if they cannot straighten their fingers.

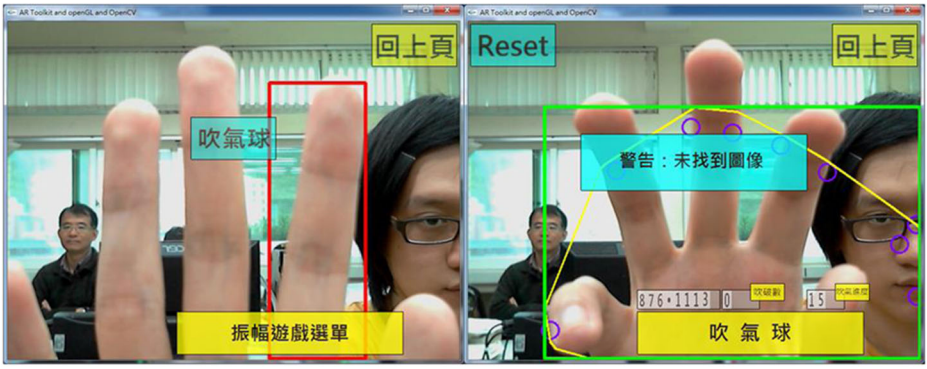
3.3 The rationale of redness test

Reddish furniture and clothes are very common in our living rooms and laboratories. Some surface pixels of these objects may be classified as skin pixels by the skin color model of [19]. For examples, the clothes and wooden patterns of part (d) of Fig. 11 are classified as skin pixels if no redness test is conducted. Thus we developed the redness test to reduce errors. At first, we randomly selected 5 millions of colors from the RGB color space and classified these colors by using the Gaussian skin color model of [19]. The distribution of those colors, classified as skin colors in the CbCr color space, is shown in part (a) of Fig. 11. The grey color pixels represents the histogram of these colors. Then the same number of colors are examined by the redness test. The colors being classified as reddish colors are recorded and rendered in the CbCr space. They form a damond-shaped scope. Their histogram is shown in part (b) of Fig. 11. By substracting reddish colors from the color space of the Gaussin model, the distribution of the colors which pass the skin pixel classification is revealed in part (c) of Fig. 11.

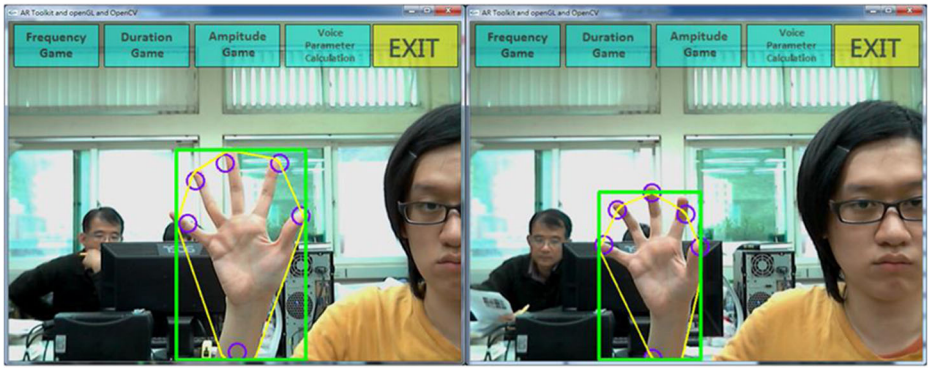
To find effective R/G and R/B ratios for the redness test, we collected images containing clothes and wooden patterns which are falsely classified as skin colors by the original skin color model. The histogram of the reddish pixels of the sample images in the CbCr space is displayed in part (e) of Fig. 11. A significant portion of these pixels are within the range of

Table 2 Hand-tracking & finger-detection rates using Leap Motion

Process \ Distances (cm)	20	25	30	60	90	100
Finger detection (straight)	100 %	100 %	100 %	0 %	0 %	0 %
Finger detection (curled)	100 %	100 %	100 %	0 %	0 %	0 %



(a) Operation distance =20 cm



(b) Operation distance =90 cm

Fig. 10 Finger detection in different distances and gestures, left images:straight hand gestures, right images: curled hand gestures. (**a**) At 20 cm, the detection rate is lower. (**b**) At 90 cm, the detection rate is 100 %. The feasible operation distance is between 30 ~ 90 cm

reddish color of the redness test if the threshold ratios q of R/B and R/G are set to 1.5. If we reduce q, some real skin pixels may be removed by the redness test. On the other hand, if q are too large, more reddish pixels of clothes and furniture are classified as skin pixels. Therefore, the ratios are set at 1.5 and these reddish pixels will mostly be filtered out by the redness test.

3.4 Comparison with commercial products

There are some commerical HCI facilities which can be used to implement AR user interfaces. We compare four commercial products with our system. The summary of our study is contained in Table 3. We selected the following aspects in the comparison: (1) capturing RGB images, (2) extra hardwares, (3) multiple user environment, (4) finger detection, and (5) operational distances. The first aspect ensures that AR user interface can be implemented. The second aspect is used to compare the price. The last three criteria reveal the feasibility for our home-based health caring system.

The four commercial products are Leap Motion [7], Xtion [8], Kinect [9], and Kinect 2 [6]. Leap Motion is utilized for capturing hand gestures. It is the only system in our study, which cannot capture RGB images. In our health caring system, RGB images of the ambient are

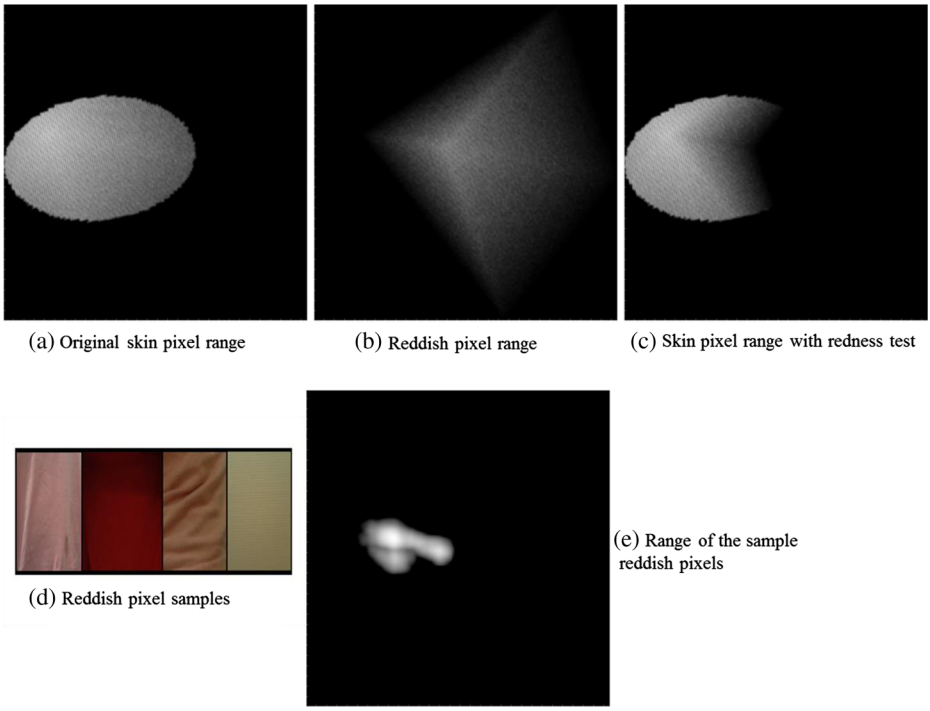


Fig. 11 Effect of redness test, (**a**) skin pixel range in the CbCr color space, (**b**) the range of reddish pixels, *R*/*G* and *R*/*B* ratios ≥1.5, (c) skind pixel range with redness test, (**d**) samples of reddish pixels, (**e**) histogram of the samples in the CbCr color space

combined with computer-generated images to display not only the AR user interface but also the progressions of uttering training games. Lacking this feature disqualifies Leap Motion for our health caring system. Users can buy a web-camera and combine the hand gestures detected by Leap Motion with the images fetched by the Web-camera. However, registration of the two streams need extra computational efforts. Among these 5 systems, ours requires minimal hardware support. Web-cameras are almost standard appendages of modern desk-top computers. Our system requires no extra hardware beside a web-camera. Thus it is almost costless.

By default, Xtion and Kinect can track only one person in the input image stream. If multiple people are to be identified in the image stream, multiple instances of the stream have to be created and different tracking skeletons have to be used. This requires extra efforts and will slow down the performance of the system. Furthermore, these two systems are developed

Table 3 System comparison based on feasibility for the health caring system

	RGB image capture	Extra hardware	Multiple user environment	Finger detection	Operation distance
Leap Motion	No	Depth camera	Yes	Yes	10–40 cm
Xtion ProLive	Yes	Depth + RGB camera	Limited	Weak	80–350 cm
Kinect	Yes	Depth + RGB camera	Limited	Weak	80–400 cm
Kinect 2	Yes	Depth + RGB camera	Yes	Yes	50–450 cm
Ours	Yes	Web-camera	Yes	Yes	30–90 cm

for tracking the bodies of the users. Their operation ranges are too far for our health caring system, though they are useful for constructing other physical therapy applications. The other problem is that their capabilities in fingertip detection are weak. Their main design goals are human gesture detection but not finger tracking. It is difficult to use fingertips as input devices in both the systems. Leap Motion is very capable of tracking hands and detect hand gestures. However, the users' hands must appear within the interaction box to guarantee successful operations [7]. Its feasible operation range is much smaller than that of our system. The operation distances of Kinect and Xtion are too far for a desk-top computer. Users may need an extra size screen to display the interface, context, and training games.

Kinect 2 supports finger detection and multiple-user environment [5]. Its operational distacne is between 50 to 450 cm. This distance may be feasible for the HCI of the health-caring system. However, in the health caring system, the distance between the users and the computer screen is usually about 80 ~ 90 cm, as shown in Fig. 9. As the users change gestures and move hands, the distance between the users' hands and the camera may be less than 50 cm. Since the feasible operational distance of Kinect 2 is beyond 50 cm, it may fail to capture the users' hands. Another problem with Xtion, Kinect and Kinect 2 is their weights. In the proposed method, the web-camera is mounted on the top edge of the screen because the weight of the web-camera is light. On the other hand, Kinect, kinetc 2 and Xtion are much heavier. They can not be directly mounted on the top edge of the screen. The users have to invest extra hardwares to support these facilities. In conclusion, to develope a halth caring system in a desk-top machine, the proposed HCI method is more feasible by weighting all the aforementioned factors.

4 Discussions and conclusion

In this work, we propose a multi-user HCI method for user interface design. In the proposed method, a color model and a redness test are used to extract skin regions at first. Then the active hand is located by using an optical flow computation and image differencing. Since we model the hand-tracking by using an FSM, the authority of interaction can be transferred among the users. Fingertips are found in the contour of the active hand by using geometrical features. The fingertips are employed as pointing devices in an AR user interface for a home-based voice caring system. The test results show that the proposed HCI method is robust, efficient and practical for the purpose.

In many HCI programs, hand gestures are used as vocabularies to compose and send signals to the computer. The success of these HCI programs relies on accurate hand gesture recognition algorithms which are difficult to design. In our voice caring system, we do not have to recognize any hand gesture. Instead, we combine AR techniques and the proposed HCI method to build a virtual user interface which is easy to use and learn. Therefore, we improve the usability of the voice health caring system and save computational resources for the voice quality diagnosis and assessment modules which are computationally intensive.

In testing the HCI method, we found that the lighting condition may cause difficulties in skin region segmentation. The skin color model adopted in our method is not suitable for extracting skin pixels in a room with darker lighting effects. As depth cameras are getting popular, the depth field of the input image may be a good cue for hand region segmentation [18]. A depth camera can also provide the speed of the movement of hand in the direction of the camera. This speed data could be useful in creating the AR user interface in our HCI method.

References

1. Azuma RT (1997) A survey of augmented reality. Presence: Teleoperators and Virtual Environments 6(4):355–385
2. Erol A, Bebis G, Nicolescu M, Boyle RD, Twombly X (2007) Vision-based hand pose estimation: a review. Comput Vis Image Underst 108(1):52–73
3. Gutzeit E, Vahl M, Zhou Z, von Lukas U (2011) Skin cluster tracking and verification for hand gesture recognition. In Proceedings of ISPA 2011, pp. 241–246
4. Hasan MM, Mishra P (2012) Hand gesture modeling and recognition using geometric features: a review. Canadian Journal on Image Processing and Computer Vision 3(1):12–26
5. http://www.engadget.com/2014/10/08/kinect-for-windows-finger-tracking/, read 2015/09/28.
6. https://dev.windows.com/en-us/kinect/hardware, read 2015/09/28.
7. https://developer.leapmotion.com/documentation/java/devguide/Leap_Coordinate_Mapping.html, read 2015/08/10.
8. https://msdn.microsoft.com/zh-tw/library/hh438998.aspx, read 2015/08/10.
9. https://www.asus.com/tw/3D-Sensor/Xtion_PRO_LIVE/specifications/, read 2015/08/10.
10. Lee B, Chun J (2009) Manipulation of virtual objects in marker-less AR system by fingertip tracking and hand gesture recognition. In Proceedings of ICIS 2009, pp. 1110–1115
11. Lee D, Lee SG (2011) Vision-based finger action recognition by angle detection and contour analysis. ETRI J 33(3):415–422
12. Letessier J, Berard F (2004) Visual tracking of bare fingers for interactive surfaces. In Proceedings of ACM UIST'04, pp. 119–122
13. Manresa C, Varona J, Mas R, Perales FJ (2005) Real-time hand tracking and gesture recognition for human-computer interaction. Electronic Letters on Computer Vision and Image Analysis 5(3):96–104
14. Kay Elemetric Corporation (2003) Multi-Dimensional Voice Program (MDVP) operations manual, Model 5105
15. Raheja JL, Das K, Chaudhary A (2011) An efficient real time method of fingertip detection. In Proceedings of TIMA-2011, pp. 447–450
16. Rekimoto J (2002) SmartSkin: an infrastructure for freehand manipulation on interactive surfaces. In Proceedings of ACM CHI 2002, pp. 113–120
17. Ueng SK, Luo CM, Chang H (2012) Voice quality assessment and visualization. In Proceeding of CISIS 2012, pp. 618–623
18. Wilson AD (2010) Using a depth camera as a touch sensor. In Proceedings of ITS2010: Devices and Algorithms, pp. 69–72
19. Wu YW, Ai XY (2008) Face detection in color images using AdaBoost algorithm based on skin color information. In Proceedings of IEEE Workshop on Knowledge Discovery and Data Mining, pp. 339–342
20. Zhou H (2009) An OPS hand tracking algorithm based on optical flow. In Proceedings of World Congress on Software Engineering, pp. 190–194
21. Zhou H, Ruan Q, Chen H (2010) A new approach of hand tracking based on integrated optical flow analysis. In Proceedings of ICSP 2010, pp. 1194–1197

Shyh-Kuang Ueng holds a PhD degree in computer science. Currently, he is an assistant professor of the Department of Computer Science and Engineering of National Taiwan Ocean University in Keelung City Taiwan. His research interests include Computer Graphics, Visualization, Medical Imaging Systems and Scientific Computing.

Guan-Zhi Chen received a master degree on Computer Science from Nation Taiwan Ocean University in 2012. His research interests include game engine design, multi-media and human computer interaction. Currently, he focuses his efforts on developing algorithms and programs for HCI and Augmented Reality. In addition, he also devotes himself to revise graphics engines for various applications.