A Long-Range Touch Interface for Interaction with Smart TVs

Jaeyeon Lee, DoHyung Kim, Jaehong Kim, Jae-il Cho, and Joochan Sohn

A powerful interaction mechanism is one of the key elements for the success of smart TVs, which demand far more complex interactions than traditional TVs. This paper proposes a novel interface based on the famous touch interaction model but utilizes long-range bare hand tracking to emulate touch actions. To satisfy the essential requirements of high accuracy and immediate response, the proposed hand tracking algorithm adopts a fast color-based tracker but with modifications to avoid the problems inherent to those algorithms. By using online modeling and motion information, the sensitivity to the environment can be greatly decreased. Furthermore, several ideas to solve the problems often encountered by users interacting with smart TVs are proposed, resulting in a very robust hand tracking algorithm that works superbly, even for users with sleeveless clothing. In addition, the proposed algorithm runs at a very high speed of 82.73 Hz. The proposed interface is confirmed to comfortably support most touch operations, such as clicks, swipes, and drags, at a distance of three meters, which makes the proposed interface a good candidate for interaction with smart TVs.

Keywords: Smart TV, touch interface, bare hand tracking, human computer interaction.

Manuscript received Oct. 19, 2011; revised Aug. 14, 2012; accepted Aug. 22, 2012.

This work was supported by the IT R&D program of MKE & KEIT [10041610, The development of the recognition technology for user identity, behavior and location that has a performance approaching recognition rates of 99% on 30 people by using perception sensor network in the real environment] and the ETRI R&D Program of KCC (Korea Communications Commission), Rpe. of Korea [11921-03001, "Development of Beyond Smart TV Technology"].

Jaeyeon Lee (phone: +82 42 860 5507, leejy@etri.re.kr), DoHyung Kim (dhkim008@etri.re.kr), Jaehong Kim (jhkim504@etri.re.kr), Jae-il Cho (jicho@etri.re.kr), and Joochan Sohn (jcsohn@etri.re.kr) are with the IT Convergence Technology Research Laboratory, ETRI, Daejeon, Rep. of Korea.

http://dx.doi.org/10.4218/etrij.12.0111.0667

I. Introduction

Smart TVs are television sets with integrated Internet connections and offer more advanced computing abilities than do conventional TVs. These devices allow users to run various applications and to search for and find videos, movies, photos, and other content on the Web. The smart TV is an evolving concept that is still expanding its boundaries. However, what matters in the context of interaction is that smart TVs are obviously far more complex to navigate than are conventional TVs.

The dominant interface for the traditional TV has been the remote control. Although modern remote controls have a lot of buttons and controls, those used by typical users are generally very limited. With such a limited interface, it is difficult to enjoy various services that smart TVs provide, such as Web browsing and game applications.

One obvious alternative to the remote control is hand gesture recognition. Many researchers have proposed the use of hand tracking and gesture recognition as a powerful and natural interface for interaction with computers, game machines, and/or television sets [1]-[6]. In particular, bare hand tracking allows the user to control devices without attaching additional devices. Furthermore, the use of the interface should be intuitive enough so that little training is required for the users. In this respect, we focus on the famous touch interface that smartphones provide. Since the introduction of Apple's iPhone several years ago, the touch interface has gained enormous popularity. Many new smartphones and tablet PCs with similar interfaces have been successively announced, and many users are becoming accustomed to the interface. In addition, the interface has been proven to be versatile enough to support hundreds of

thousands of applications.

In this paper, a novel interface that mimics the interaction model of a touch interface is proposed. However, in the proposed interface, bare hand tracking is used to emulate a touch action, which eliminates the need to actually touch the screen. That is, the tracked hand position is interpreted as a coordinate on the screen, and the touch is enabled when the hand is pushed forward.

The interface should satisfy several requirements to be used for interaction with smart TVs. Television is typically enjoyed in a relaxing environment, such as in the user's own living room. Therefore, the interface should allow as much freedom as possible for the user. The user most likely controls the TV sitting on a sofa. Thus, the interface should work at a distance of at least three meters. Furthermore, we cannot restrict the sitting position of the user, which prohibits the use of zoom cameras. With a typical camera with a horizontal field of view of 60 degrees, the hand blob is as small as 20×20 pixels, even when a relatively high resolution of 640×480 is used, which imposes quite a challenge in visual tracking. Also, clothing that exposes a lot of skin may cause problems in color-based trackers.

Another important problem is the user's inability to accurately point at a specified point on the screen. When the user is repeatedly requested to point to a designated point on the screen with his hand, the actual hand position may vary significantly each time, which inherently limits the pointing accuracy, irrelevant of the recognition performance. To cope with this problem, it is necessary to incorporate the idea of a "human in the loop" [1]. As the recognition result is reflected in the graphical user interface (GUI), the user can adjust his own movement to achieve his original intention, which will be reflected in the GUI again. In this scenario, the feedback loop must be fast enough to guarantee a sense of intuitiveness as an appreciable delay may confuse the user. It has been pointed out that the response time should be less than 50 ms [7]. That is, the bare hand tracking algorithm should run faster than 20 Hz. Also, the tracking should be accurate and stable enough for the user to point at a button as small as a key on a screen keyboard. Finally, to emulate the touch action, it is essential to detect the 3D information of the hand.

Naturally, the usability of the interface heavily depends upon the performance of the hand tracking algorithm. Many efficient algorithms for this purpose have been reported upon [7]-[15]. However, considering the aforementioned constraints, the available options are quite limited. First, the hands in the image are very small and often blurry due to fast motion, which makes it difficult to use sophisticated model-based approaches [2], [8], [9], [15], [16]. A hard time constraint also forces us to adopt simpler algorithms. Still, it is necessary to preserve the tracking accuracy to guarantee excellent user experiences.

Considering these constraints, a hand tracking algorithm based on color information is adopted, which is known to be very efficient in the context of computational complexity [10], [12], [17]-[19]. However, it is also known that color-based trackers have many drawbacks.

In this paper, an online color modeling technique and the integration of motion and color information are adopted to cope with the drawbacks, which will be described in section II in detail. In addition to the base algorithm presented in section II, several ideas are proposed to solve the problems often encountered in hand tracking. The ideas are tested on a database that consists of 98 video files that captured the pointing activities of 11 subjects. Based on the experiment results, the proposed ideas are adopted or discarded to construct the best hand tracking algorithm.

The experiment results show that the ratio of erroneous frames (the distance between a recognized hand position and the ground truth is larger than 10 pixels) is 1.86% and the ratio of seriously erroneous frames (the above distance is larger than 20 pixels) is 0.60%. Although these error rates do not directly correspond to the quality of user experience, the small numbers are very encouraging. In addition, the proposed interface is applied to several GUIs, including a smartphone emulator, verifying that most operations of a touch interaction model, such as clicking, swiping, or dragging, can be supported comfortably from afar.

This paper consists of five sections. Section II describes the system configuration and introduces the proposed base algorithm. Section III discusses the proposed ideas for further improvements over the base algorithm. In section IV, the ideas are tested on the aforementioned database to verify the merits and demerits of the proposed ideas. Finally, section V ends this paper with some concluding remarks.

II. Proposed Algorithm

1. Obtaining 3D Information

In the proposed interface, the touch action is emulated by pushing the hand forward. Thus, it is indispensable to capture 3D information of the hands, which requires appropriate sensor devices.

There are several approaches to obtain 3D information. Active depth cameras have been receiving a lot of attention since the recent introduction of Kinect from Microsoft. The sensor captures the distance information very reliably with a resolution of 640×480 and is available at a relatively low cost compared to that of other depth cameras [20], [21]. However the cost is still more than ten times that of general CMOS

cameras, which is a critical factor in mass production. The bulkiness of the sensor also hinders its adoption, considering the extremely slim design of modern TV sets.

Another popular approach is stereo matching. By using two cameras separated by an appropriate distance, a dense depth map that contains the distance information of the scene can be obtained. However, it is well known that stereo matching often fails when textureless regions are dominant [22]. Also, the stereo matching demands heavy computation, which makes adoption difficult without the support of specialized hardware.

Considering the above alternatives, we use dual cameras in a configuration similar to stereo matching. However, instead of trying to generate a dense depth map, only the 3D information of the points of interest is obtained. In the proposed interface, the points of interest are the center of the face and the hand being tracked. To achieve this purpose, two independent hand trackers track the face and hand regions in respective images. Triangulation is then applied to calculate the distance to the region [22].

Because hand trackers do not depend on the regional matching, a lack of texture or large occlusion by a long baseline does not cause serious problems. Also, the hand tracker is far faster than the full-fledged stereo matching process. This approach has another benefit in that rigorous calibration and rectification is not mandatory, which allows the interface to work with any general web cameras placed in an appropriate configuration. In the proposed interface, two 640×480 images are used, considering the distance at work.

2. Person Finder

Figure 1 shows the overall flow of the proposed interface. As shown in the figure, the initial step is to find the person in the scene. Once a person is detected, the control is transferred to the hand tracker. While the hand tracker tracks the hand, it also verifies the person's existence. If the hand tracker loses the person, control is returned to the person finder to repeat the process.

Finding a person in a scene is a very popular research topic. The methods range from simple ones that use the skin colors to more sophisticated ones that detect trained patterns of the face, head, and shoulders or patterns of the full body [23]-[27].

The proposed person finder basically depends on the face detector, considering the high reliability of this technology [28]. Although the face detector may fail if the frontal view of the face is not accessible, it is reasonable to assume that the person can face the camera if she wants to interact.

Although the face detector is quite efficient in itself, applying it to two high-resolution images is still time consuming. To accelerate the process, a frame difference image is analyzed to narrow down the candidate region. If a face is detected in this region, the control is transferred to the hand tracker with the rectangular face information.

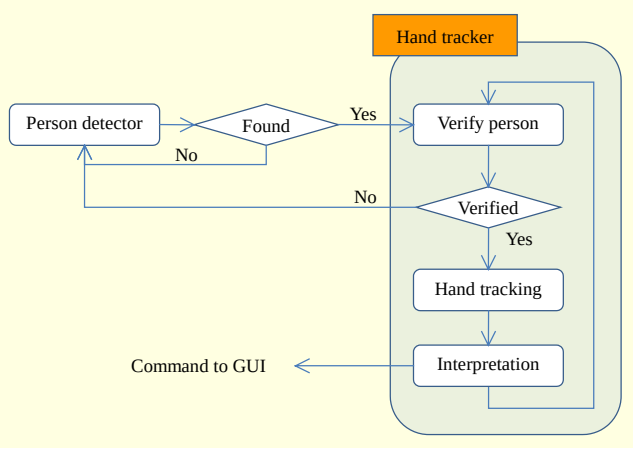


Fig. 1. Overall flow of proposed interface.

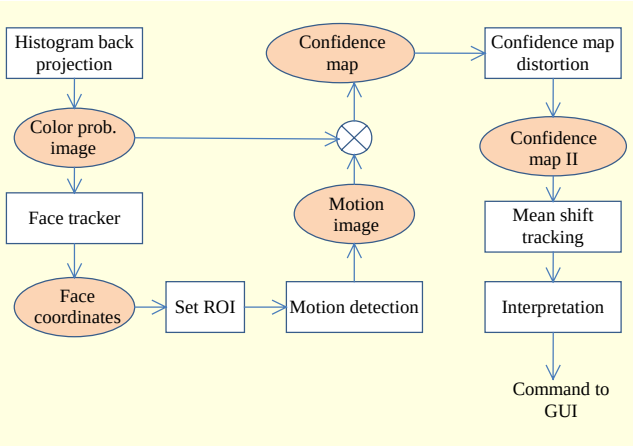


Fig. 2. Overall flow of hand tracker.

3. Hand Tracker

A. Face Tracking

Figure 2 shows the overall flow of the hand tracker. For every input image, the hand tracker tracks the face first. The purpose of face tracking is to maintain contact with the interacting user. The system must know if the user moves or leaves the scene, as the hand coordinates are calculated relative to the face location. The face location is also important to determine the region of interest (ROI) in the image. Once the face location and size is known, the range of hand motion is restricted by human anatomy. In the proposed interface, the ROI is determined proportionally to the face region, as shown in Fig. 3 (for left-handed users, the ROI rectangle is shifted accordingly to the right).

For the face tracking, a mean shift algorithm is applied on a

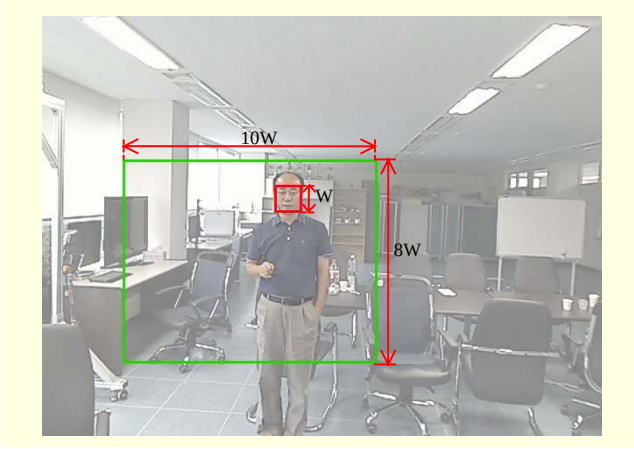


Fig. 3. ROI.

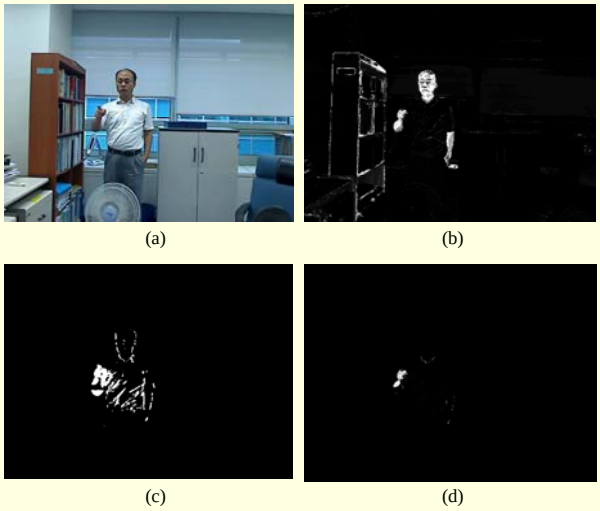


Fig. 4. Combining color and motion information: (a) input image, (b) color probability, (c) binary motion image, and (d) confidence map.

color probability image, which is a gray image brighter in the facial colored region and darker in other regions. One of the main drawbacks in color-based tracking is the sensitivity to illumination changes [10], [19], [29]. Good skin color detectors often fail if the illumination is different from that of the modeling. This problem can be solved using online color modeling because the model and input data are under the same conditions. This approach also has the benefit of handling skin color differences among the world's population. In this paper, a histogram back projection is adopted for this purpose [30], [31], which defines the color probability image as follows:

$$P_i = 255 \cdot \min(\frac{M_i}{I_i}, 1), \quad (1)$$

where P_i is a pixel value of the probability image for the i-th bin, M_i is the object histogram value, and I_i is the whole image histogram for the i-th bin.

This color probability image is later reused for the hand tracking, which is also based on the assumption that the hand color is closely related to the color of the face. In addition to this color tracking, a face detector is applied to the tracked region once every N frames for verification, where N is a constant determined by considering a balance between the computational cost and reliability. When verification by the face detector fails for a certain number of consecutive trials, the system decides that it has lost the person and yields the control to the person finder. On the contrary, when the verification succeeds, the skin color is remodeled so that the recent illumination condition can be reflected.

B. Motion Detection

Another major drawback for color-based tracking is processing background regions bearing similar colors to the skin [10], [11]. However, in the proposed interface, a more serious problem is caused by exposed skin areas other than the hand. The purpose of the hand tracker is to track the hand position. Thus, it is disastrous if the tracker happens to follow the face or exposed arms. If the confidence map is only based on the color characteristics, such improper tracking is inevitable, as the face and arm areas are made up of pixels reflecting the person's actual skin color. To cope with this problem, motion information is utilized under the assumption that the hand may have a stronger tendency toward motion than other areas where skin is likely to be exposed.

Figure 4(b) shows an example of a color probability image for the input of Fig. 4(a). In addition to the hand region, nearby bookshelves and the face region are represented by bright pixels, which indicate a high probability of skin. Considering that we are mainly interested in a moving hand, a binary motion image, shown in Fig. 4(c), is logically ANDed with the probability image resulting in the final confidence map of Fig. 4(d). As shown in the figure, only hand regions are highlighted, as expected. By adopting this approach, the final confidence map will have higher values when the object with skin color is moving fast and will be zero if no motion is detected. In this approach, the confidence map becomes a zero image when the user temporarily stops moving. In this case, the results of the previous frame are preserved, which is reasonable because the hand has remained still. Also, this approach is helpful to cope with the stability problem caused by oscillatory motions [7].

C. Finding Hand

Before the tracking starts or once the tracker loses the hand, it is necessary to find the initial hand position to track. For hand detection, the given confidence map is divided into grids of a constant size, and the pixel values for each grid are summed to

find the grid with maximum probability. The grid with the maximum summation value is a candidate for the hand region; however, this is true only if the summation value exceeds a predetermined threshold. A small summation value indicates that there is no moving skin-colored region. In this case, the hand finding is declared as a failure, and the system waits for the next image frame to repeat the process.

When a probable region is found, the mean shift tracker is applied with the maximum grid region as the initial search position. Here, the initial hand size is assigned with the same as that of the face.

D. Tracking Operation

Hand tracking is performed by applying mean shift tracking on the above-mentioned confidence map. Once the tracking is over, the density of the region (mean value of the pixels in the region) is evaluated if it exceeds a predetermined threshold. Too small a density indicates either a failure of tracking or a temporary stop of user motion. A tracking failure occurs when the hand moves too fast and the new hand position goes beyond the search range of the mean shift tracker, which is not detectable in this stage and will be described later. The latter case is an expected situation in which the coordinates of the previous frame are preserved.

E. Coordinate Transformation

In this paper, we are working with three coordinate systems. The first uses the image coordinates that we have dealt with. These are the hand coordinates in the given image. The second is world coordinates that can be calculated using triangulation from the two image coordinates obtained from the respective hand tracker. The real world position of the face is used as the origin in this coordinate system. Finally, the world coordinate should be transformed into an appropriate screen coordinate that the user wants to point toward.

The transformation from an image coordinate into a world coordinate is a trivial problem only if both hand trackers detect the hand successfully. If one or both of the hand trackers fails to generate valid image coordinates, this frame is simply ignored.

Unlike an image-to-world coordinate transformation, a world-to-screen coordinate transformation is not straightforward. The mapping basically depends upon the user's pointing behavior, which is different from person to person. Thus, a good coordinate transformation requires an extensive analysis of user behaviors. However, the "human in the loop" also plays an important role in this context. If a reasonable mechanism is provided, users can adapt quite easily.

In the proposed interface, a virtual screen is assumed to be in front of the user, as shown in Fig. 5. There is no *a priori* rule to determine the position and size of the virtual screen, which should be determined through user testing. Generally speaking, too small a virtual screen restricts the resolution of the pointing, while too large a screen forces excessively large movements for the users, both of which are not desirable. Several positions and sizes of the virtual screen were tested to find the appropriate configuration.

Fig. 5. Virtual screen in front of user.

Once the virtual screen is determined, the hand position on the virtual screen is linearly mapped to the screen coordinates, and, if the hand intersects with the virtual screen, the touch status is declared.

F. Filtering

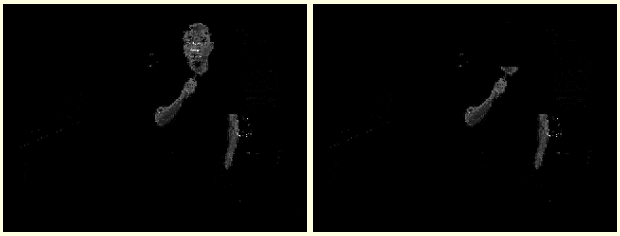
The screen coordinates often show precarious motions due to noise in the image processing procedures, which makes the user very nervous. For stabilization, a Kalman filter is applied to the 2D coordinates.

Touch status recognition also suffers similar instability. Hence, the status is decided by the threshold (distance from the face to the virtual screen), and it can oscillate rapidly when the hand distance is near the threshold level. To avoid this problem, a status change is declared only if the status change is confirmed for a certain number of consecutive frames.

G. Click Lock

In the proposed interface, a touch is emulated by pushing the hand forward. However, when the user pushes her hand forward, not only the distance but also the plane coordinates are inevitably affected. This may make the user very confused, resulting in an uncontrollable interface. Even if the user pushes his hand when the cursor is on the button, the plane coordinate may have changed at the time when the touch is recognized. This dissonance also occurs when the touch is released by pulling the hand back. To solve this problem, the user's

intended position just before the pushing action starts is estimated by analyzing the history. Based on the assumption that the plane coordinates may vary more rapidly if the pushing action starts, we search for the point where the local coordinate variation becomes minimum from the recent coordinate history. The plane coordinate is then locked to the detected coordinate until the touch is released or the recognized plane coordinate is changed over a certain threshold.



(a) (b)

Fig. 6. Resetting face region: (a) color probability image and (b) reset face region.

III. Proposed Improvements

As described in the previous section, online color modeling and integration of motion solves many problems inherent in color-based trackers. However, the proposed tracker still suffers from errors, especially relating to exposed skin areas other than those of the hands. In this section, several ideas are proposed to cope with the problems usually encountered in the given situation. Also, by applying the ideas to the database, which will be described later, we will verify the usefulness or uselessness of the ideas used to construct the best hand tracking algorithm.

In a visual analysis, individually dealing with specific cases generally does not produce desirable results, due to too many variations in real situations. The proposed ideas, rather, try to maneuver the confidence map with the intention to achieve the desired effect.

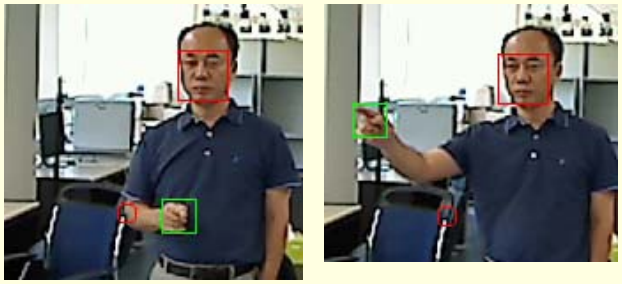
1. Resetting Face Region in Confidence Map

Unlike the background, the face moves, although generally not fast. That is, there are possibilities that the face region has higher values in the confidence map, which may cause the tracker to be stuck in this region. To avoid such errors, the face region of the confidence map is reset to zero, as shown in Fig. 6, before the mean shift tracker is applied.

2. Intentional Distortion of Confidence Map

When the user moves her hand to point at the screen, her arm also moves. Furthermore, her arm region also has skin color characteristics if she is wearing a sleeveless shirt, which may be frequent when in her living room. In this case, there is no way to discriminate the hand from the arm region. To cope with this problem, the confidence map is intentionally distorted such that the hand region has higher values than the arm region has. To do so without a complicated structural analysis, it is assumed that the hand is the farthest region from the elbow, and the position of the elbow is simply estimated according to the position and size of the face, as follows:

$$x = \mathrm{f}.left - \mathrm{f}.Width(), \tag{2}$$



(a) (b)

Fig. 7. Estimated elbow position: (a) correct and (b) incorrect estimation.

$$y = \mathrm{f}.bottom + \mathrm{f}.Height() \cdot \frac{11}{5}, \tag{3}$$

where f is the rectangle of the face region and Width() and Height() are the functions that return the width and height of the rectangle (for left-handed users, (2) is modified accordingly).

The red circle in Fig. 7 is the estimated elbow position. Although, this formula does not estimate the elbow position correctly, as in the case of Fig. 7(b) in which the hand region is still farther than the arm from the estimated point. To distort the confidence map, the filter in Fig. 8 is used, whose value is determined by a sigmoid function of the following equation:

$$f = Sigmoid(\frac{d-50}{16}), \tag{4}$$

where f is the filter value and d is the distance from the center.

The filter is designed to have a minimum value of about 0.1 at the center, and the value is increased as the distance to the elbow position increases. The value saturates at 1.0 if the distance exceeds 150. Figure 9 shows the results of the confidence map distortion. As intended, the arm region darkens while the hand remains as is.

3. Representation of Motion Information

When a homogeneous object moves, frame differencing

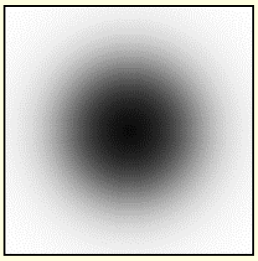


Fig. 8. Distortion filter.

(a) (b)

Fig. 9. Distortion of confidence map: (a) before distortion and (b) after distortion.

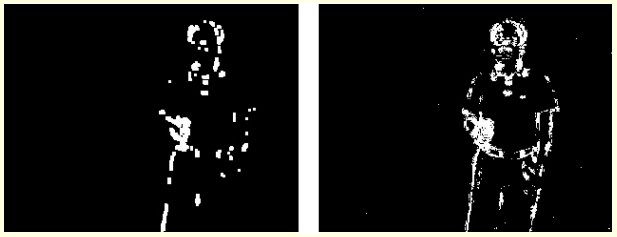
roughly detects the boundary of the region instead of the whole object, which may result in a loss of information in the confidence map. The simplest way to cope with this problem is to dilate the motion image, as shown in Fig. 10(a). Although this may overly extend the moving regions, it may not be a problem unless a skin-colored background object exists in the immediate vicinity because it will be ANDed with the color probability image.

Another solution is to accumulate the difference images for several frames, as shown in Fig. 10(b). This approach has a benefit when a hand comes to a stop. If the motion decreases, the confidence map has a very small number of bright points, which makes the confidence map less reliable. Using the accumulated difference image has a smoothing effect on the temporal axis.

4. Cross-Checking

An obvious redundancy exists because two cameras are capturing the identical scene simultaneously. The redundancy can be utilized to either accelerate the processing or to cross-check the correctness of the results. To use the information for computational efficiency, the results of the one hand tracker may be applied to shrink the search space of the other tracker. However, the mean shift tracker is already doing the search space pruning and is sufficiently fast. Thus, we use the information to check the correctness of the results.

If the dual camera is positioned horizontally, the disparity in the vertical direction is theoretically zero if the cameras are correctly calibrated and rectified. Although rigorous calibration and rectification are not assumed in the proposed configuration, the disparity in the vertical direction still may be small and nearly constant. Hence, if the vertical disparity changes abruptly, we can decide that one or both of the hand trackers has failed. Although there is no way to correct the result, it is possible to use the information to improve the interaction. There are two cases of such errors. One is temporary and is automatically corrected in the subsequent frames, which can be dealt with by simply ignoring the results of the erroneous frame. Another case is when the tracker is stuck on an erroneous region, such as the face or arm. In this case, the error is possibly propagated to the subsequent frames. To deal with the latter case, the confidence value of the hand tracking information is decreased. If the confidence value drops under the threshold, the tracker abandons the tracking information and lets the hand finder take control. In the former case in which the tracker corrects itself, the confidence value is recovered so that the tracker returns to its normal status.



(a) (b)

Fig. 10. Representation of motion: (a) dilation and (b) accumulation.

IV. Experiments

For a collection of realistic motion data, 4×3 numbered buttons are drawn on a screen, and the subjects are instructed to point and click the button with a number randomly generated by the program. A human operator decides if the subject's clicking operation is completed and then proceeds to the next number until 20 clicking operations are completed. This user action is captured at a rate of 30 fps (frames per second), using dual cameras. Thus, the video length is not a constant and ranges from one minute to two minutes. From 11 subjects, 49 such video pairs are collected, resulting in 114,460 frames in total. For all the frames, the regions of the face and the hand being tracked are annotated by a human operator and are used as a ground truth. In the evaluation, instead of calculating the average distance between the tracked position and the ground truth, two levels of erroneous frames are counted because the

Table 1. Performance comparison.

Algorithm	0000	0001	0010	0011	0100	0101	0110	0111
Error level 1	7.70%	4.71%	5.76%	3.88%	5.25%	2.68%	3.66%	**1.86%**
Error level 2	5.59%	2.78%	3.72%	2.01%	3.12%	0.79%	2.28%	**0.60%**
Algorithm	0000	1001	1010	1011	1100	1101	1110	1111
Error level 1	11.03%	5.76%	8.15%	4.96%	7.67%	3.00%	5.98%	2.21%
Error level 2	8.37%	3.61%	5.63%	2.80%	5.34%	0.94%	4.29%	0.79%

Table 2. Contribution of each idea.

	Applied		Discarded		Improvement	
	Error 1	Error 2	Error 1	Error 2	Error 1	Error 2
Reset face	6.09%	3.87%	4.44%	2.58%	–37.31%	–50.12%
Distortion of confidence map	4.04%	2.67%	6.49%	4.27%	37.82%	37.59%
Difference representation	4.56%	2.81%	5.97%	3.78%	23.75%	25.58%
Cross checking	3.63%	1.69%	6.90%	4.75%	47.38%	64.39%

exact position of the ground truth depends upon the subjective decision of the human workers. In the first error level, the distance between the tracked position and ground truth is larger than 10 pixels, which is a slight deviation and is mostly recovered in the following frame. In the second error level, the frames have an error distance of larger than 20 pixels. These are more serious errors and often indicate that the tracker has lost the hand or is tracking an object other than the target hand. In this case, chances are that the erroneous frames are propagated to the subsequent frames.

Using the proposed algorithm described in section II as a basis, the four ideas described in section III are selectively applied to form sixteen algorithms, which are applied to the database to evaluate the above-mentioned two levels of error rates. The experiment results are shown in Table 1. Each digit of the algorithm number indicates whether the ideas of section III are applied.

- 1st digit: Face region of the confidence map is reset or not reset.
- 2nd digit: Distortion of the confidence map is applied or not applied.
- 3rd digit: Motion is represented by the accumulated frame difference or by a dilated frame difference.
- 4th digit: Cross-checking is applied or not applied.

Compared to the base algorithm (algorithm 0000, whose first level error rate is 7.70% and second level error rate is 5.29%), the improved algorithms generally show a higher performance, culminating in algorithm 0111 with a 1.86% error rate in the first level and a 0.60% error rate in the second level.

To verify the contribution of each idea, the algorithms are grouped by the idea. That is, among the sixteen algorithms, eight adopt a certain idea while the other eight do not. For each idea, both algorithm groups are evaluated, as shown in Table 2. As the table shows, three of the ideas are helpful in making improvements, whereas resetting the face region deteriorates the accuracy. Although the idea can effectively prevent the tracker from sticking to the face region, it also inherently prohibits correct tracking when the moving hand is occluded by the face region. The experiment results show that the problem continues.

For the purpose of comparison, a pure color-based tracker [31] is applied to the same experiment resulting in 17.20% and 13.69% for each level of error respectively, which is far inferior to the base algorithm. Although the context of the reported algorithm is different, this result shows that tracking small objects like hands in a cluttered background is difficult to do with color information only.

In addition to the high accuracy, an immediate response is also indispensable for a natural interaction. The proposed algorithm can run at 82.73 Hz, simultaneously processing two 640×480 images (the test images are preloaded to exclude the I/O time). Although the processing speed is measured on a high-end PC with a 3.33-GHz quad core, the speed far exceeds that of 20 Hz, easily supporting the cognitive requirements of the user [7].

The experiment described above proves that the proposed hand tracking algorithm performs superbly. However, the algorithm does not guarantee excellent user experience in interacting with smart TVs. To verify the usability of the proposed interface, it is applied to a smartphone emulator.

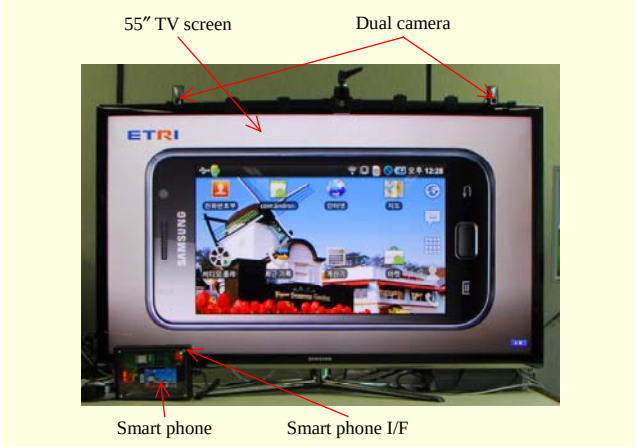


Fig. 11. Smartphone emulator.

As shown in Fig. 11, a real smartphone is connected to the TV screen. The screen of the smartphone is intercepted to be displayed on the TV screen, and the user's hand movement is interpreted as a touch command and is transferred to the smartphone. Although this device has no practical purpose, we can verify that most of the smartphone operations can be accomplished by the proposed interface. It is confirmed that we can use the smartphone with its inherent touch interface, including swiping the pages, launching applications, and dragging slide bars. It is also possible to use the screen keyboard for word search input, which demonstrates the high accuracy in processing pointing.

V. Conclusion

Smart TVs are expected to change our everyday lives. Although recent discussions have mainly focused on smart TV services, this paper addressed the problem of an interaction mechanism, the solution of which will be crucial to ensure pleasant user experiences.

Considering that smart TVs are not only a broadcasting medium but also a computing platform, a touch interaction model was adopted in the proposed interface. However, in lieu of actually touching the screen, the touch operation was emulated using bare hand tracking, allowing the interface to work from afar.

The essential part of the interface is high-performance hand tracking. In this paper, a novel hand tracking algorithm based on the combination of color and motion was proposed, and, by introducing additional improvements, an excellent hand tracking performance was achieved. The proposed interface proved to work robustly, even when several other regions bear a similar color to the hand. It also takes only about 12 ms to process two 640×480 images.

To verify the usability of the proposed interface as a control for smart TVs, a smartphone emulator was implemented. Using the smartphone emulator from a position three or more meters away from the screen was found to be sufficiently comfortable. This is a good indication that the proposed interface can be adopted for interaction with smart TVs.

Most operations of the touch interaction model, such as clicking, swiping, and dragging, were accomplished by the proposed interface. However, multitouch, which is often used for zooming, has not yet been implemented. In further research, the proposed hand tracking algorithm is expected to accommodate dual hand tracking and to support a multitouch interface.

References

[1] W.T. Freeman et al., "Computer Vision for Interactive Computer Graphics," *IEEE Comput. Graphics Appl.*, vol. 18, no. 3, 1998, pp. 42-53.

[2] C.C. Wang and K.-C. Wang, "Hand Posture Recognition Using Adaboost with SIFT for Human Robot Interaction," *LNCIS*, vol. 370, 2008, pp. 317-329.

[3] Y. Verdie, B. Fang, and F. Quek, "MirrorTrack-Tracking with Reflection-Comparison with Top-Down Approach," *Proc. Int. Conf. Multimodal Interfaces Workshop Mach. Learning Multimodal Interfaces*, 2009, pp. 347-350.

[4] K. Smith et al., "Real-Time 3D Hand Tracking in a Virtual Environment," *Proc. SPIE: Int. Society Optical Eng.*, vol. 5006, 2003, pp. 529-543.

[5] C. Manresa et al., "Real-Time Hand Tracking and Gesture Recognition for Human-Compute Interaction," *Electron. Lett. Comput. Vision Image Anal.*, 2000.

[6] L. Sun, U. Klank, and M. Beetz, "EYEWATCHME: 3D Hand and Object Tracking for Inside Out Activity Analysis," *IEEE Conf. Comput. Vision Pattern Recog.*, 2009, pp. 9-16.

[7] C. Hardenberg and F. Berard, "Bare-Hand Human-Computer Interaction," *Proc. ACM Workshop Perceptive User Interface*, 2001.

[8] P. Garg, N. Aggarwal, and S. Sofat, "Vision Based Hand Gesture Recognition," *World Academy Sci., Eng., Technol.*, vol. 49, 2009, pp. 972-977.

[9] M. Gorce, N. Paragios, and D.J. Fleet, "Model-Based Hand Tracking with Texture, Shading and Self-Occlusions," *Proc. IEEE Conf. Comput. Vision Pattern Recog.*, 2008, pp. 1-8.

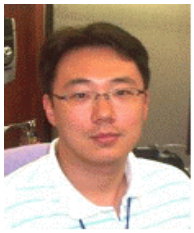
[10] M. Yuan et al., "Robust Hand Tracking Using a Simple Color Classification Technique," *Int. J. Virtual Reality*, vol. 8, no. 2, 2009, pp. 7-12.

[11] F. Mahmoudi and M. Parviz, "Visual Hand Tracking Algorithms," *Proc. Geometric Modeling Imaging New Trends*, 2006, pp. 228-232.

[12] C. Manders et al., "Robust Hand Tracking Using a Skin Tone

and Depth Joint Probability Model," *8th IEEE Int. Conf Autom. Face Gesture Recog.*, 2008.
[13] J. Romero et al., "Dynamic Time Warping for Binocular Hand Tracking and Reconstruction," *Proc. IEEE Int. Conf. Robotics Autom.*, 2008, pp. 2289-2294.
[14] D. Chik, J. Trumpf, and N.N. Schraudolph, "3D Hand Tracking in a Stochastic Approximation Setting," *LNCS*, vol. 4814, 2007, pp. 136-151.
[15] W.Y. Chang, C.S. Chen, and Y.P. Hung, "Appearance-Guided Particle Filtering for Articulated Hand Tracking," *Proc. IEEE Comput. Soc. Conf. Comput. Vision Pattern Recog.*, vol. 1, 2005, pp. 235-242.
[16] A. Argyros and I. Lourakis, "Binocular Hand Tracking and Reconstruction Based on 2D Shape Matching," *Proc. Int. Conf. Pattern Recog.*, 2006, pp. 207-210.
[17] V. Vezhnevets, V. Sazonov, and A. Andreeva, "A Survey on Pixel-Based Skin Color Detection Techniques," *Proc. GraphiCon*, 2003, pp. 85-92.
[18] R. Kjeldsen and J. Kender, "Finding Skin in Color Images," *Proc. Int. Conf. Autom. Face Gesture Recog.* , 1996, pp. 312-317.
[19] Y. Wu and T.S. Huang, "Color Tracking by Transductive Learning," *Proc. IEEE Comput. Soc. Conf. Comput. Vision Pattern Recog.*, 2000, pp. 133-138.
[20] S.B. Gorturk, H. Yalcin, and C. Bamji, "A Time-Of-Flight Depth Sensor: System Description, Issues and Solutions," *Comput. Vision Pattern Recog. Workshop*, 2004.
[21] K.L. Boyer and A.C. Kak, "Color-Encoded Structured Light for Rapid Active Ranging," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 9, no. 1, 1987, pp. 14-28.
[22] S.T. Barnard and W.B. Thompson, "Disparity Analysis of Images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 2, no. 4, 1980, pp. 333-340.
[23] P. Viola and M. Jones, "Robust Real-Time Object Detection," *IEEE ICCV Workshop Statistical Comput. Theories Vision*, Vancouver, Canada, July 2001, pp. 1-25.
[24] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," *IEEE CVPR*, June 2005, pp. 886-893.
[25] M. Hussein, F. Porikli, and L. Davis, "A Comprehensive Evaluation Framework and a Comparative Study for Human Detectors," *IEEE Trans. Intell. Transp. Syst.*, vol. 10, no. 3, Sept. 2009, pp. 417-427.
[26] M. Enzweiler and D.M. Gavrila, "Monocular Pedestrian Detection: Survey and Experiments," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 12, Dec. 2009, pp. 2179-2195.
[27] C. Zeng and H. Ma, "Robust Head-Shoulder Detection by PCA-Based Multilevel HOG-LBP Detector for People Counting," *IEEE ICPR*, Aug. 2010, pp. 2069-2072.
[28] B. Jun and D. Kim, "Robust Real-Time Face Detection Using Face Certainty Map," *Int. Conf. Biometrics*, vol. 4642, no. 125, Aug. 2007, pp. 29-38.
[29] L. Sigal, S. Sclaroff, and V. Athitsos, "Skin-Color Based Video Segmentation Under Time-Varying Illumination," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 7, 2004, pp. 862-877.
[30] M.J. Swain and D.H. Ballard, "Color Indexing," *Int. J. Computer Vision*, vol. 7, no. 1, 1991, pp. 11-32.
[31] J.I. Agbinya and D. Ress, "Multi-Object Tracking in Video," *Real-Time Imaging*, vol. 5, no. 5, 1999, pp. 295-304.

Jaeyeon Lee received his PhD from Tokai University, Tokyo, Japan, in 1996. He has been a research scientist at ETRI, Daejeon, Rep. of Korea, since 1986. His research interests include robotics, pattern recognition, computer vision, and biometric security.



DoHyung Kim received his PhD from Pusan National University, Busan, Rep. of Korea, in 2009. He has been a research scientist at ETRI, Daejeon, Rep. of Korea, since 2002. His research interests include human-robot interaction, computer vision, and pattern recognition.

Jaehong Kim received his PhD from Kyungpook National University, Daegu, Rep. of Korea, in 2006. He has been a research scientist at ETRI, Daejeon, Rep. of Korea, since 2001. His research interests include socially assistive robotics for elder care, human-robot interaction, and gesture/activity recognition.

Jae-il Cho received his BS and MS in electrical engineering from Sungkyunkwan University, Suwon, Rep. of Korea, in 1990 and 1992, respectively. Since joining ETRI, Daejeon, Rep. of Korea, in 1992, he has carried out research on communication networks and intelligent service robot systems. His main research interests include embedded systems, ASIC design, and robot control.

Joochan Sohn received his MS from Hankuk University of Foreign Studies, Seoul, Rep. of Korea, in 1990. He has been a research scientist at ETRI, Daejeon, Rep. of Korea, since 1996. His research interests include intelligent agents for robots and human-robot interaction, and he mainly focuses on multimodal interfaces for human-robot interaction.