International Journal of Innovative Computing, Information and Control
Volume **9**, Number **2**, February **2013**

ICIC International ©2013 ISSN 1349-4198
pp. 851–864

TOUCHLESS USER INTERFACE BASED ON MARKER DETECTION AND TRACKING FOR REAL-TIME MOBILE APPLICATIONS

Il-Lyong Jung[1], Nikolay Akatyev[2], Won-Dong Jang[1]
Leonardo Juniti Nomoto[2] and Chang-Su Kim[1]

[1]School of Electrical Engineering
Korea University
145, Anam-ro, Seongbuk-gu, Seoul 136-701, Korea
illyong@korea.ac.kr; wdjang@mcl.korea.ac.kr; cskim.ieee.org@gmail.com

[2]Mobile Communications
LG Electronics
Seoul, Korea
nikolay.akatyev@lge.com; junitileo@gmail.com

Received December 2011; revised May 2012

Abstract. *A touchless user interaction system based on marker detection and tracking is proposed for real-time mobile applications. The proposed algorithm can robustly estimate users' control information with a camera on a mobile device, which often has limited hardware resources and is influenced by varying environmental conditions. First, we detect a pre-registered marker based on the normalized correlation coefficient. Then, we track the marker motion by employing the contrast invariant mean-shift algorithm. More specifically, the proposed contrast invariant mean-shift algorithm transforms a candidate frame in order to match its histogram into the histogram of the target frame. It then tracks feature points by performing the mean-shift on both original and transformed candidate frames adaptively. To evaluate the performance of the proposed interaction system, we implement 'Painting', 'Camera' and 'Virtual Keypad' applications. Experimental results demonstrate that the proposed algorithm provides better interaction performance than the conventional method, while demanding lower computational complexity and thus supporting real-time user interaction.*

Keywords: User interaction, Mobile user interface, Touchless user interface, Contrast invariant tracking, Pattern analysis

1. Introduction.

Mobile devices have evolved from simple phones into powerful multimedia creators and players with recent advances in mobile and multimedia technologies. In 2009, 4.6 billion people used mobile phones, corresponding to 67 percent of the global population [1]. Mobile phones play a significant part in the human lifestyle, in which people can listen to music, take photographs, play games, watch digital video contents, and control consumer electronic devices [2]. Nowadays, most mobile phones include at least VGA displays, integrated digital cameras, and high-clock-rate CPUs with dedicated graphics processing units (GPUs) to support these applications. These components empower mobile phones to become sophisticated computing centers. However, people need intuitive user interfaces to control applications on mobile phones more easily. The design of user interfaces significantly affects the usability of mobile phones [3].

Traditional user interaction systems for mobile phones have been controlled by 4×3 telephone keypads with letter mapping [4]. This remains the commonest user interface for making a phone call and sending a message. However, the keypad interface has a limitation that it occasionally requires multiple typing for a single task. Though the

QWERTY keypad has been employed to prevent multiple typing, it is impractical for delicate interaction, such as drawing a curve. Voice interaction [5] also has been used for dialing or giving commands, instead of typing key buttons to complete the task. However, voice interaction is inappropriate for giving multiple commands quickly due to the reaction delay. Sensor-based interaction, which detects the vibration, tilt, or motion of a device, has been used [6, 7, 8]. Though this interaction system is suitable for specific tasks, e.g., pivoting displays, it cannot support precise interaction. Touch-based user interfaces have been developed to support more precise interaction. A user can control situation-specific tasks by touching a screen. However, since touch-based interfaces require additional hardware for touch screens, they increase handset prices and are not suitable for low-end mobile phones. Moreover, they are useless in some situations, for example, when fingers are wet or users wear gloves.

Computer vision techniques have been used for camera-based user interfaces to overcome the above mentioned issues and provide users with novel user experiences [9, 10, 11, 12, 13, 14]. To acquire control information for camera-based user interfaces, color detection methods [9, 10, 11, 12] and object tracking methods [13, 14] can be used. Especially, in [15, 16, 17, 18, 19], camera-based interfaces have been proposed for mobile phones. In [15], a motion-based method was used to measure the translation and rotation of a device. In [16], a user interaction method using the Kalman filter tracking was presented. However, these methods require too high complexity to be employed in practical applications. In [17], Drab and Artner developed a low complexity system based on scene analysis. However, it yields inaccurate control when scenes contain repetitive texture or they are captured under varying contrast conditions. Bulbul et al. [18] proposed an interaction system based on face tracking. Gallo et al. [19] adopted a finger tracking method, which estimates the gradients of the red chrominance to update new positions. These approaches [18, 19] support real-time user interaction, but they assume that there is only a single object with a uniform skin color in a scene.

In this paper, we develop a touchless user interaction system based on marker detection and tracking for real-time applications on mobile phones. To support mobile devices with limited hardware resources in varying environmental conditions, we first register a user-defined marker pattern. Then, we detect the pattern based on the normalized correlation coefficient [20, 21], and track it using the proposed contrast invariant mean-shift algorithm to extract control information under varying contrast and scale conditions. Specifically, to achieve reliable tracking under varying illumination and exposure conditions, the contrast invariant mean-shift algorithm first transforms a candidate frame using the sorted histogram specification [22], so that its histogram is matched to the histogram of the target frame. Then, we track feature corner points by performing the mean-shift on both original and transformed candidate frames. The detected marker motion is then used to control applications. Extensive simulation results demonstrate the proposed user interaction system can be employed for various applications efficiently.

The remainder of the paper is organized as follows. Section 2 describes the overall structure of the proposed interaction system and explains the marker detection and the contrast invariant mean-shift tracking techniques. Sections 3 and 4 describe applications and implementation issues of the proposed system, respectively. Section 5 discusses the performance of the proposed system. Finally, Section 6 concludes the paper.

2. **Proposed System.** Figure 1 shows the architecture of the proposed user interaction system, which uses video information from a camera as the input. The proposed system requires an activation procedure, which turns on the camera, so as to operate the camera module at users' requests only. In this work, the activation is performed by

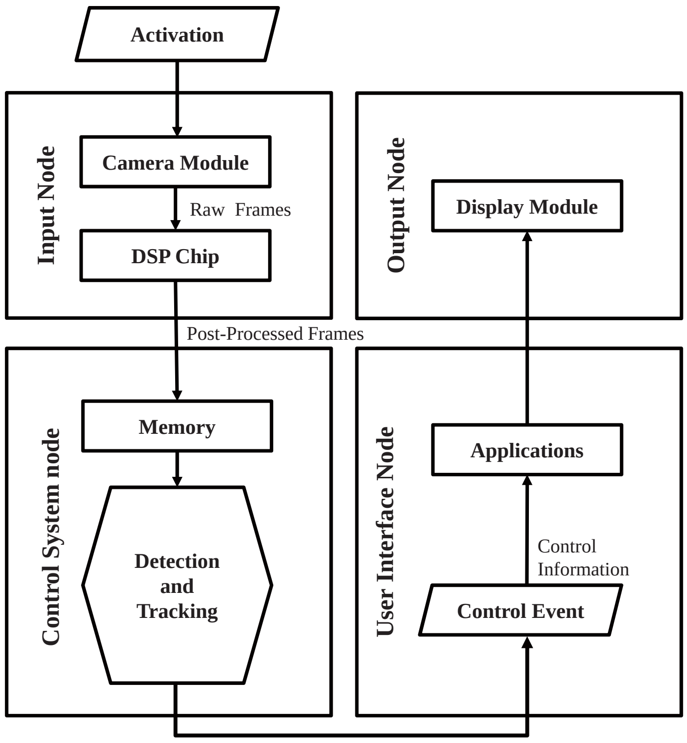


FIGURE 1. The overall architecture of the proposed user interaction system

touch-based user interaction, which enables the system to initiate the input node. However, the proposed system can be initiated by any user interaction techniques, such as voice interaction.

When the system is activated, the camera module captures a scene to generate raw frames. The mobile device then processes the raw frames using a DSP chip to adjust their qualities. Then, the post-processed frames are stored in the memory buffer. By detecting and tracking the marker in the frames, the control system node analyzes events. This information is then transferred to the user interface node. Finally, applications can be controlled. At the same time, the captured frames and application data are also shown on the display module.

2.1. **Control system node.** When designing user interaction systems for mobile devices, several constraints should be taken into account [23]. Major constraints are related to limited resources, including low CPU power and small memory. Moreover, mobile devices have mobility-related constraints, since input frames from mobile cameras have widely varying characteristics, such as brightness change, span and tilting. In this work, the control system node has the detection module and the tracking module, which analyze input frames from the camera in varying environments. Therefore, we also have the resource-related constraints and the mobility-related constraints.

To overcome these constraints, we use a pre-registered marker, which is shown in Figure 2. Although the proposed system can be utilized with any marker, a marker with indistinctive features, such as bare hands, may lead to poor performance or require high system complexity. Therefore, we utilize the marker in Figure 2 in this work, which has distinctive color and edge features.

2.1.1. *Detection module.* Let M be the marker image, which is pre-registered as shown in Figure 2. Let I be an input image, captured by the camera. Let $I(x, y)$ and $M(x, y)$

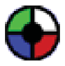
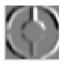
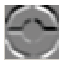
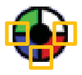
Pattern	Grayscale	Gradient (hor.)	Gradient (ver.)	Feature point
				

FIGURE 2. A user-defined marker pattern and its horizontal and vertical gradient components

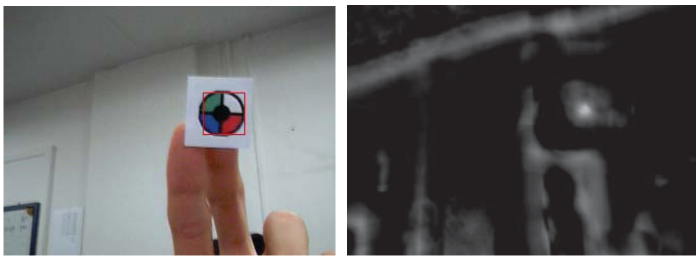


FIGURE 3. (a) An input image and (b) its normalized correlation coefficient map

denote the values of pixels at position (x, y) in the images I and M, respectively. In the detection module, we detect the marker in the input image I. The lighting and exposure conditions of the image vary greatly in mobile environments. Therefore, we detect the matching block in I, which is most similar to the marker image M, by employing the normalized correlation coefficient [20, 21], as illustrated in Figure 3. We can prevent matching errors due to different lighting and exposure conditions based on the normalization. The normalized correlation coefficient is defined as

$$r(x,y) = \frac{\sum_{x',y'}\left[\hat{M}(x',y')\hat{I}(x+x',y+y')\right]}{\sqrt{\sum_{x',y'}\left[\hat{M}(x',y')\right]^2 \sum_{x',y'}\left[\hat{I}(x+x',y+y')\right]^2}} \tag{1}$$

where $\hat{M}(x',y') = M(x',y') - \bar{M}$, $\hat{I}(x+x',y+y') = I(x+x',y+y') - \bar{I}$, $\bar{M}$ is the average value of $M(x',y')$, and $\bar{I}$ is the average value of $I(x+x',y+y')$, respectively. The normalized correlation coefficient $r(x,y)$ ranges from -1 to 1. In the input image I, we search the best matching block with the largest coefficient. If the coefficient is less than a pre-described threshold, we carry out the detection in the next frame. This continues until we find the block with a coefficient larger than the threshold.

2.1.2. *Tracking module based on the contrast invariant mean-shift algorithm.* The detection module searches the best matching block in I. However, the detection requires relatively high complexity to search the marker over the entire frame. Since mobile devices have the resource-related constraints, it is necessary to minimize the complexity of the matching for real-time processing. Also, due to the mobility-related constraints, the intensity similarity-based matching often fails to achieve accurate tracking when matching pixels exhibit different intensities due to varying contrast conditions. Moreover, since the distance between the camera and the marker changes frequently, we should detect the marker robustly regardless of the scale change of the marker. To this end, we propose the contrast invariant mean-shift algorithm, which is based on the mean-shift technique [13, 14] and the sorted histogram specification [22]. Figure 4 shows the flowchart of the proposed contrast invariant mean-shift algorithm.

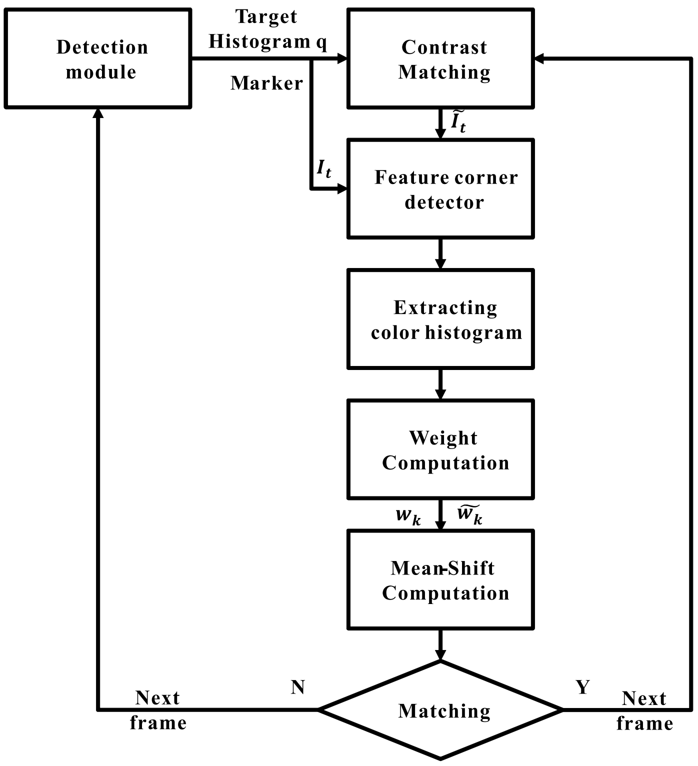


Figure 4. The flowchart of the tracking module

We employ the mean-shift technique, which is a simple and reliable approach to real-time object tracking. However, the conventional mean-shift tracking is not robust against the scale change of the marker as well as varying contrast conditions. Therefore, we first transform each candidate frame to compensate for the contrast variations based on the sorted histogram specification, which we proposed in our preliminary work [22]. A histogram represents the probability distribution of pixels in an image, and it is widely used to specify image characteristics [24]. Since neighboring frames in a captured video are highly correlated, their histograms should be similar to each other. Let L_{t-1} and L_t denote the discrete random variables, representing pixel values in the target frame I_{t-1} and the candidate frame I_t, respectively. Also, let $h_{L_{t-1}}(l_{t-1})$ and $h_{L_t}(l_t)$ denote the probability distribution functions for the target frame I_{t-1} and the candidate frame I_t. Then, the cumulative distribution functions (CDF's) $H_{L_{t-1}}$ and H_{L_t} are defined as

$$H_{L_{t-1}}(l_{t-1}) = \Pr\{L_{t-1} \le l_{t-1}\} = \sum_{k=0}^{l_{t-1}} h_{L_{t-1}}(k) \tag{2}$$

$$H_{L_t}(l_t) = \Pr\{L_t \le l_t\} = \sum_{k=0}^{l_t} h_{L_t}(k)$$

The CDF values $H_{L_{t-1}}(l_{t-1})$ and $H_{L_t}(l_t)$ represent the ranks of pixels with color levels l_{t-1} and l_t, respectively, in the neighboring frames. Based on the CDF's, we transform the color levels in the candidate frame I_t to match the CDF of the target frame I_{t-1}.

However, it may cause ambiguity in the matching when multiple pixels in a candidate or target frame have the same CDF value. To resolve the ambiguity, we employ the sorted histograms [22] by employing the tie-breaking rule based on the average values of the neighboring pixels. Using the sorted histograms, we obtain the transition vectors, which transfer the histogram $H_{L_{t-1}}(l_{t-1})$ of the target frame I_{t-1} to the histogram $H_{L_t}(l_t)$ of the candidate frame I_t. Then, we acquire the transformed candidate frame $\tilde{I}_t$ using those transition vectors.

Even after we overcome the effect of varying contrast conditions, there is still a problem due to the scale changes of the marker. The distance between the camera and the marker changes in mobile environments. Therefore, the tracking method should detect the scale changes of the marker and track the marker robustly regardless of the scale changes. Although the scale-invariant feature transform (SIFT) [25] or the speeded up robust features (SURF) [26] can be used to achieve the scale invariance, they are computationally too demanding for mobile applications. Instead, we extract distinctive features from the marker and track each region separately.

Let q denote the color histogram of the matching block I_q, which was found in the detection module, and q_k denote the histogram value at level k. Similarly, let p and $\tilde{p}$ denote the color histograms of the candidate block I_p in the original candidate frame I_t and the candidate block $\tilde{I}_p$ in the transformed candidate frame $\tilde{I}_t$, respectively. Let p_k and $\tilde{p}_k$ denote the corresponding histogram values at level k.

First, we extract feature corner points of the matching block I_q. We obtain the gradient vector field of I_q. The gradient magnitude $|\nabla I_q|$ is given by

$$|\nabla I_q| = \sqrt{(S_x * I_q)^2 + (S_y * I_q)^2} \tag{3}$$

where S_x and S_y are the Sobel masks in the horizontal and vertical directions. Then, we select the three points with the largest gradients. Around each selected point, we set the feature block $B_{i,q}$, $i = 1, 2, 3$, which is depicted by a yellow rectangle in the last column in Figure 2. Each feature block is then tracked separately by the mean-shift tracking algorithm.

Let $p(x_i, y_i)$ and $\tilde{p}(x_i, y_i)$ be the histograms of the candidate blocks $B_{i,p}$ and $B_{i,\tilde{p}}$ at position (x_i, y_i) in the original candidate frame I_t and the transformed candidate frame $\tilde{I}_t$, respectively. For example, in case of $p(x_i, y_i)$, we iteratively track the marker and find the best position (x_i, y_i) that maximizes the similarity between the target histogram q_i of the feature block $B_{i,q}$ and the candidate histogram $p(x_i, y_i)$. The similarity between the two histograms is measured by the Bhattacharyya coefficient [27]

$$\rho(x_i, y_i) = \sum_{k=0}^{m} \sqrt{p_k(x_i, y_i) q_{i,k}} \tag{4}$$

where m is the number of histogram bins, which is set to 255 in this work.

Given the location (x_i, y_i) of the marker in the target frame, we track its location (x'_i, y'_i) in the candidate frame I_t and the transformed candidate frame $\tilde{I}_t$, respectively, so that the Bhattacharyya coefficients are maximized. To reduce the complexity, instead of computing the coefficient for each candidate location directly, we use the mean-shift rule [13]. The proposed contrast invariant mean-shift algorithm employs both the original candidate frame I_t and the transformed candidate frame $\tilde{I}_t$ to alleviate the effect of contrast variations. To combine the information in these two frames properly, we compare the similarities between the target frame and these candidate frames and then assign the weight values adaptively. Specifically, we assign the weight α to the original candidate frame I_t and the weight β to the transformed candidate frame $\tilde{I}_t$, based on the histogram intersection [28],

$$\alpha = \frac{\sum_{k=0}^{255} \min(q_k, p_k)}{\sum_{k=0}^{255} q_k}, \qquad \beta = \frac{\sum_{k=0}^{255} \min(q_k, \tilde{p}_k)}{\sum_{k=0}^{255} q_k}. \tag{5}$$

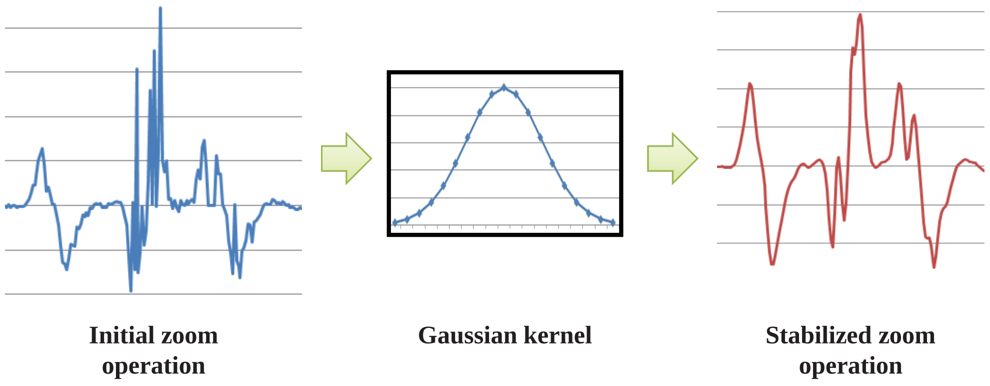


FIGURE 5. Sensitivity control using the Gaussian moving average filter

Then, using the weights α and β, we perform the contrast invariant mean-shift tracking via

$$\begin{bmatrix} x_i' \\ y_i' \end{bmatrix} = \frac{\sum_k \left\{ \alpha w_k(x_i, y_i) \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \beta \tilde{w}_k(x_i, y_i) \begin{bmatrix} x_i \\ y_i \end{bmatrix} \right\}}{\sum_k \left(\alpha w_k(x_i, y_i) + \beta \tilde{w}_k(x_i, y_i) \right)} \tag{6}$$

where

$$w_k(x_i, y_i) = \sqrt{\frac{q_{i,k}}{p_k(x_i, y_i)}}, \qquad \tilde{w}_k(x_i, y_i) = \sqrt{\frac{q_{i,k}}{\tilde{p}_k(x_i, y_i)}}. \tag{7}$$

This iterative rule is repeatedly applied until the convergence, which usually takes five to six repetitions.

2.1.3. *Click module.* In any user interaction system, the click interface is necessary to enable users to perform their desired actions. Therefore, to support the touchless user interaction more intuitively, we develop the click module based on the analysis of motion vectors. A single click operation consists of pushing the button and releasing it. In our interface, the pushing and the releasing correspond to the zoom-in and the zoom-out of the marker, respectively.

To detect the zoom-in/out operations, we compute the center position (x_c', y_c') of the three feature blocks. Then, the zoom magnitude of the ith feature block is defined as

$$|Z(x_i', y_i')| = \sqrt{(x_i' - x_c')^2 + (y_i' - y_c')^2}. \tag{8}$$

If the zoom magnitude $|Z(x_i', y_i')|$ of the tracked marker is bigger than the zoom magnitude $|Z(x_i, y_i)|$ of the target marker, the zoom-in operation is detected. Conversely, if the zoom magnitude $|Z(x_i', y_i')|$ of the tracked marker is smaller, the zoom-out operation is detected. However, if we directly detect zoom-in/out operations, false clicks may occur. To prevent this problem, we implement a sensitivity control mechanism for the click module. Specifically, we perform the stabilization using the Gaussian moving average filter as shown in Figure 5. In other words, we use

$$\bar{Z}_t(x_i', y_i') = \sum_{k=t-2}^{t+2} w(k) Z_k(x_i', y_i') \tag{9}$$

where the Gaussian kernel is $w(\cdot) = \{5/49, 12/49, 15/49, 12/49, 5/49\}$. In this way, we can implement the click module more reliably.

3. **Applications.** As shown in Figure 6, we implement three applications 'Painting', 'Camera' and 'Virtual Keypad' to test our user interface system.

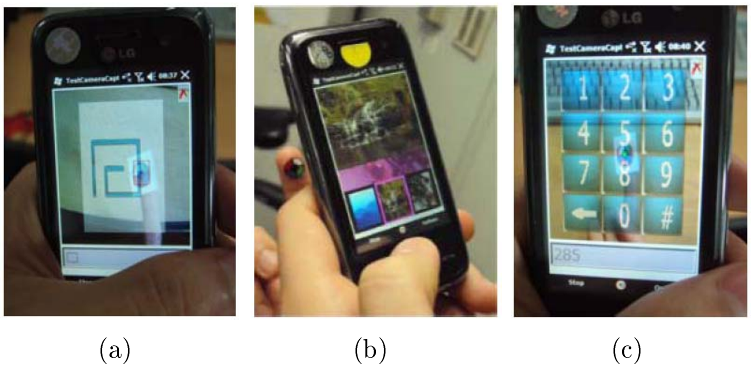


FIGURE 6. Applications of the proposed user interaction system: (a) 'Painting', (b) 'Camera' and (c) 'Virtual Keypad'

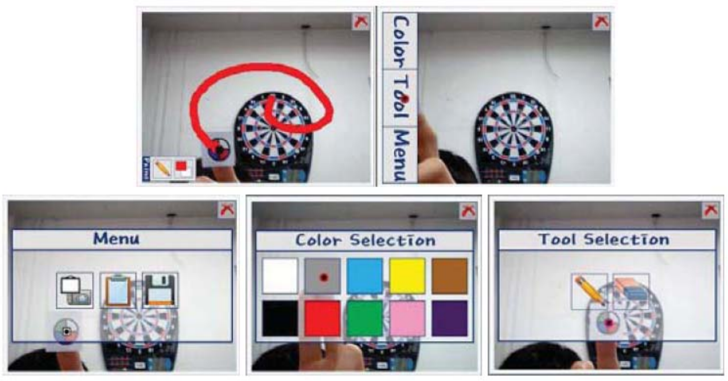


FIGURE 7. 'Painting' application and its menu

3.1. **Painting application.** Painting applications are rarely supported by keypad user interfaces or accelerometer user interfaces, since it is hard to control drawing tools precisely via those interfaces. On the contrary, touch-based user interfaces and the proposed system can support painting applications effectively. However, whereas touch-based interfaces acquire user inputs through expensive touch screen panels, the proposed system requires only a camera, which most mobile phones are equipped with. In other words, the proposed system can be employed in low-end mobile phones without requiring additional hardware.

We implement the 'Painting' application, which serves several functions with a menu: choosing a color in the color selection, drawing and erasing in the tool selection, opening and storing files in the menu selection, as shown in Figure 7. A user can draw freely by moving a finger in front of the camera. A hidden menu appears when the user moves the marker to the left. The user operates painting functions with this menu. In addition, we support several options, such as drawing on a blank background, a stored image, or a video stream. Therefore, it can offer new user experiences, such as drawing pictures during video conferences, e.g., drawing a funny face over a person at the other end of the line, writing a schedule, and marking a path on a map, as shown in Figure 8. Moreover, the 'Painting' application can be generalized to more powerful contents creation tools.

3.2. **Camera application.** Our 'Camera' application provides general functionality of taking a picture, generating a photo thumbnail, and storing or deleting an image, as shown in Figure 9. A user can take a picture, when the camera menu is selected. The application also provides the thumbnails of user's photographs in a parallel layout. These

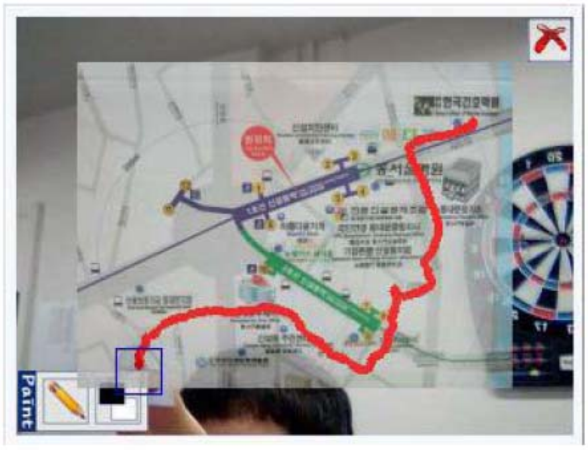


FIGURE 8. Drawing a path on a map in the 'Painting' application

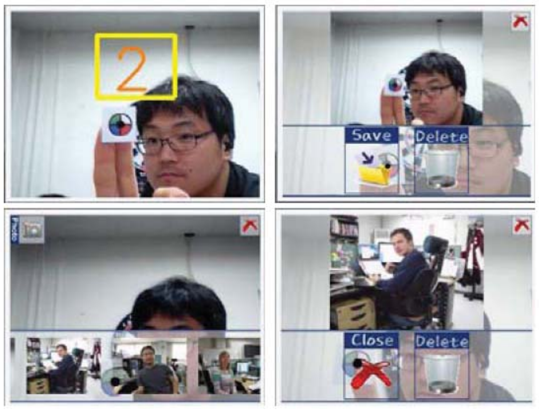


FIGURE 9. 'Camera' application and its menu

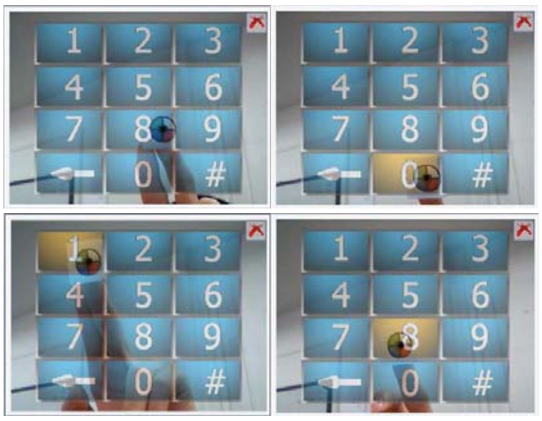


FIGURE 10. 'Virtual Keypad' application

thumbnails can be dragged to change the positions, sorted, or deleted. If a user selects a picture, the thumbnail is enlarged to the original size and can be further processed.

3.3. **Virtual keypad application.** A main function of a mobile phone is to dial phone numbers. The 'Virtual Keypad' application is implemented to support dialing or sending messages, with a virtual keypad on the screen. A user can replace the traditional keypad by a touchless virtual keypad as shown in Figure 10.

4. Implementation for Mobile Devices.

To implement the proposed user interaction system and its applications on the Windows Mobile platform, we use two computer languages: unmanaged native C++ and managed C♯. Unmanaged native C++ codes are directly compiled into machine level instructions. On the other hand, managed C♯ codes are interpreted by a just-in-time compiler, which converts the program into native codes in the Common Language Runtime (CLR) and then executes the codes. Therefore, managed codes are slower than unmanaged native codes. However, managed codes support an object oriented model and useful functions, such as garbage collection for automatic memory management.

In this work, we implement the camera capture system and the control system in unmanaged native codes to access a mobile camera efficiently. On the contrary, we develop the user interfaces for the 'Painting', 'Camera' and 'Virtual Keypad' applications, in

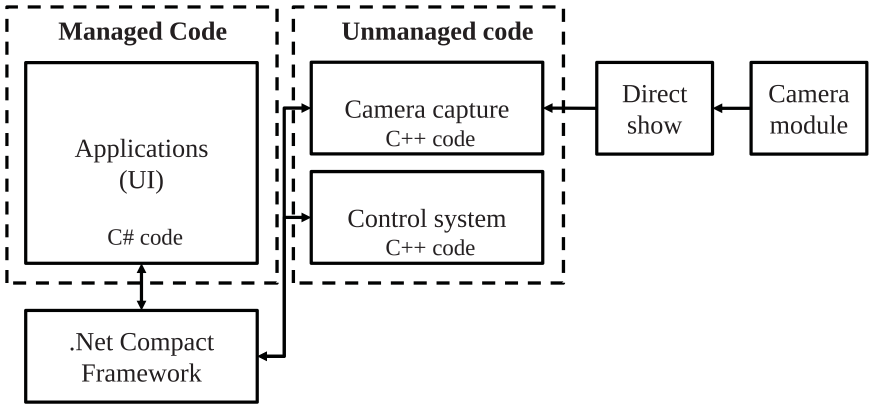


Figure 11. Software architecture of the proposed user interaction system

managed codes, which are easier to develop. We combine the system using the .NET framework [29] to achieve the inter-operability between the unmanaged native C++ codes and the managed C♯ codes, as shown in Figure 11.

5. Main Results.

We evaluate the performance of the proposed user interaction system, which is implemented in two versions: the PC version and the mobile version. In comparison with PCs, mobile devices have limited hardware resources. In addition, a camera in a mobile device is not generally fixed. The comparative tests between the PC version and the mobile version clarify the influence of the constraints in mobile devices.

We conduct the experiments of the PC version using a computer with a Pentium IV 2.4GHz processor and 2GB RAM. The mobile version is implemented on a GSM/HSDPA phone, which has a MSM7201A 528MHZ processor, 256MB RAM, a 3.0 inch color display with 240×400 pixels, and a 5 mega pixel camera. Its operating system is the Windows Mobile 6.5. The proposed system is implemented in C++ only in the PC version, but C♯ is also used in the mobile version as described in Section 4. The input frame size is QVGA (320×240) for both PC and mobile versions. The frame rate is fixed to 30 frames/s, since we target real-time applications.

5.1. Control performances.

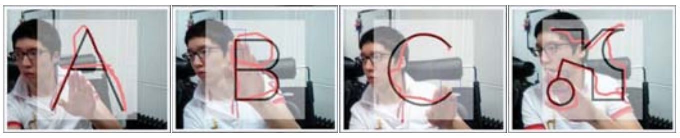
We compare the control performance of the proposed system with that of the conventional method [19]. The conventional method detects user interactions based on the skin color. We evaluate the interaction accuracy using the 'Virtual Keypad' and 'Painting' applications. We also evaluate the performance under two different lighting conditions, since camera-based methods are influenced by the lighting conditions significantly. One is the bright condition in Figure 12(a) with an external light source, and the other is the dark condition in Figure 12(c) without the light source. All experiments are performed 30 times and the average accuracy is reported.

Table 1 compares the recognition accuracy rates in the 'Virtual Keypad' application, which evaluates the average success rate when entering a given number 30 times. The proposed system provides better accuracy rates than the conventional method in both PC and mobile versions. Also, since the mobile device has limited hardware, its performance is inferior to that of the PC version.

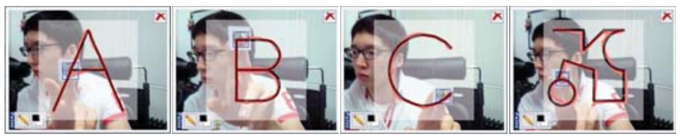
Table 2 compares the root mean square error (RMSE) performance, which measures the error rates for drawing four shapes (A, B, C, and a complex line drawing) in the 'Painting' application. The RMSE performance is computed by the square root of the average square distance between original and user-drawn routes. The proposed system provides lower RMSE performance than the conventional method. This is because the

Table 1. Comparison of recognition accuracy rates in the ‘Virtual Keypad’ application. In each test, the left number denotes the number of successes out of 30 trials, and the right number is the corresponding accuracy rate.

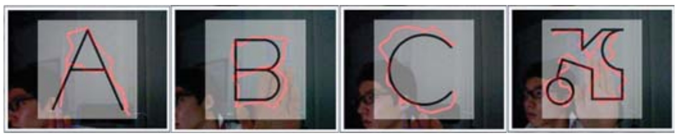
Tests	Conditions	Conventional Method [19]		Proposed Algorithm	
PC	Bright	27	90.00%	29	96.67%
	Dark	25	83.33%	28	93.33%
Mobile	Bright	25	83.33%	28	93.33%
	Dark	22	73.33%	27	90.00%



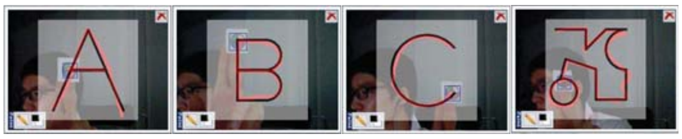
(a)



(b)



(c)



(d)

Figure 12. Control performance tests of the painting application in the PC version: (a) the conventional method in the bright condition, (b) the proposed algorithm in the bright condition, (c) the conventional method in the dark condition, and (d) the proposed algorithm in the dark condition.

conventional method becomes inaccurate if objects have similar colors, as shown in Figures 12 and 13. In addition, whereas the conventional method is negatively influenced by dark environments, the proposed system yields consistent and reliable RMSE performance in both bright and dark conditions. Also, note that for both the proposed system and the conventional method, the PC version achieves better accuracy than the mobile version, since the mobile device has limited hardware. Figures 12 and 13 show exemplar results of the PC version and the mobile version, respectively. We see that the proposed system supports more accurate control and provides better performance than the conventional method in both bright and dark conditions.

TABLE 2. Comparison of RMSE performances of the proposed system and the conventional method [19] in the 'Painting' application

Test	Condition	Algorithm	Shape				Average
			A	B	C	Complex line drawing	
PC	Bright	Conventional method	5.27	5.67	5.31	6.18	5.61
		Proposed algorithm	3.65	3.43	2.35	3.46	3.22
	Dark	Conventional method	6.90	7.01	7.92	8.57	7.60
		Proposed algorithm	4.29	3.88	2.97	3.93	3.77
Mobile	Bright	Conventional method	5.75	5.70	5.81	6.07	5.83
		Proposed algorithm	3.81	3.71	2.94	4.34	3.70
	Dark	Conventional method	8.74	7.90	8.19	8.71	8.39
		Proposed algorithm	4.23	4.23	3.21	4.84	4.13

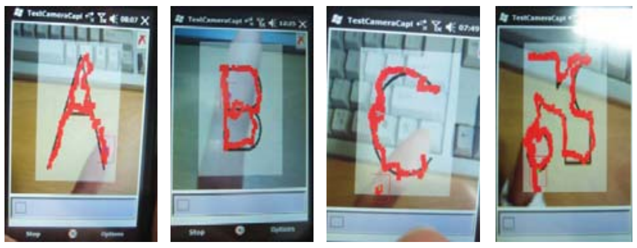
5.2. **Computational complexity.** The computational complexity of the proposed interaction system is low enough to be used for mobile devices. It operates at 30 frames/s and achieves real-time processing. Conversely, the conventional method achieves 17 frames/s only, since it requires heavy computations to differentiate between multiple skins colored objects.

These simulation results indicate that the proposed system provides better control performance than the conventional method, whilst requiring lower computational complexity.

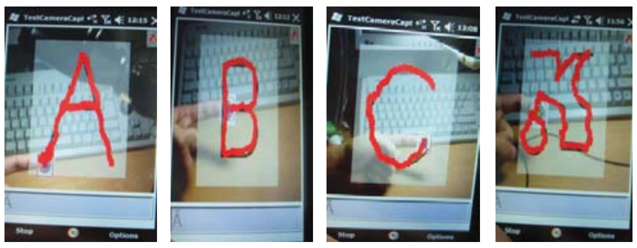
6. Conclusions.

We proposed an efficient touchless user interaction system for real-time applications in mobile devices. The proposed system can estimate control information accurately in diverse environments by detecting and tracking a marker based on the contrast invariant mean-shift algorithm. It was shown that the proposed system can support various applications, including the 'Painting', 'Camera' and 'Virtual Keypad'. The proposed system is suitable for mobile devices, since it can achieve real-time processing without degrading interaction quality. Therefore, the proposed user interaction system can overcome device-related and user-related constraints and offer novel user experiences. Simulation results demonstrated that the proposed algorithm provides better control performance than the conventional method [19], whilst requiring lower computational complexity.

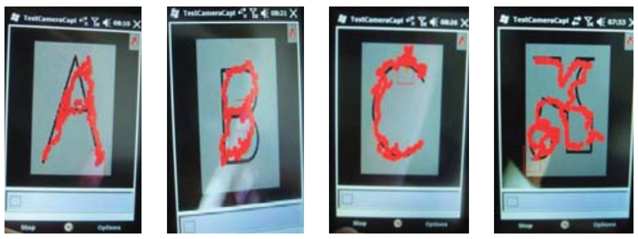
Acknowledgment. This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MEST) (No. 2012-011031), and partly by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (No. 2012-0000916). The authors also gratefully thank Victoria Bondarchuk and Tae Hyun Kim at the Mobile Handsets R&D Center, LG Electronics for their invaluable comments.



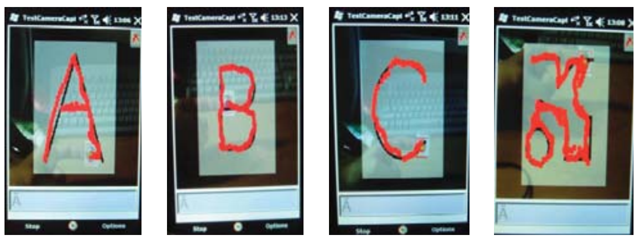
(a)



(b)



(c)



(d)

FIGURE 13. Control performance test in the painting application in the mobile version: (a) the conventional method in the bright condition, (b) the proposed algorithm in the bright condition, (c) the conventional method in the dark condition, and (d) the proposed algorithm in the dark condition.

REFERENCES

[1] International Telecommunication Union, *Measuring the Information Society 2010*, Geneva, 2010.

[2] T. Capin, K. Pulli and T. Akenine-Möller, The state of the art in mobile graphics research, *IEEE Computer Graphics and Applications*, vol. 28, no.4, pp.74-84, 2008.

[3] R. P. Morris and J. J. Tomlinson, A mobile device user interface with a simple, classic design, *IEEE Trans. Consumer Electron.*, vol.54, no.3, pp.1252-1258, 2008.

[4] ITU-T E.161, *Arrangement of Digits, Letters and Symbols on Telephones and Other Devices That Can Be Used for Gaining Access to a Telephone Network*, 2001.

[5] J. Iso-Sipilä, M. Moberg and O. Viiki, Multi-lingual speaker-independent voice user interface for mobile devices, *Proc. of IEEE ICASSP*, 2006.

[6] S. Strachan and R. Murray-Smith, Muscle tremor as an input mechanism, *Proc. of ACM Symp. User Interface Software and Technology*, 2004.

[7] J. Rekimoto, Tilting operations for small screen interfaces, *Proc. of ACM Symp. User Interface Software and Technology*, pp.167-168, 1996.

[8] J. Baek and B.-J. Yun, A sequence-action recognition applying state machine for user interface, *IEEE Trans. Consumer Electron.*, vol.54, no.2, pp.719-726, 2008.

[9] E. Hjelmas and B. K. Low, Face detection: A survey, *Computer Vision and Image Understanding*, vol.83, no.3, pp.236-274, 2001.

[10] J. Brand and J. S. Mason, A comparative assessment of three approaches to pixel-level human skin-detection, *Proc. of IEEE ICPR*, pp.1056-1059, 2000.

[11] T. S. Caetano, S. D. Olabarriaga and D. A. C. Barone, Performance evaluation of single and multiple-Gaussian models for skin color modeling, *Proc. of XV Brazilian Symp. Computer Graphics and Image Processing*, 2002.

[12] P. Kakumanu, S. Makrogiannis and N. Bourbakis, A survey of skin-color modeling and detection methods, *Pattern Recognition*, vol.40, no.3, pp.1106-1122, 2007.

[13] D. Comaniciu, V. Ramesh and P. Meer, Kernel-based object tracking, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol.25, no.5, pp.564-577, 2003.

[14] C. Zhao, A. Knight and I. Reid, Target tracking using mean-shift and affine structure, *Proc. of IEEE ICPR*, 2008.

[15] M. Rohs, Real-world interaction with camera phones, *Proc. of Int'l Symp. Ubiquitous Computing Systems*, 2004.

[16] J. Hannuksela, P. Sangi and J. Heikkila, A vision-based approach for controlling user interfaces of mobile devices, *Proc. of IEEE CVPR*, 2005.

[17] S. A. Drab and N. M. Artner, Motion detection as interaction technique for games & applications on mobile devices, *Proc. of Pervasive Mobile Interaction Devices Workshop at Pervasive Conference*, 2005.

[18] A. Bulbul, Z. Cipiloglu and T. Capin, A face tracking algorithm for user interaction in mobile devices, *Proc. of Int'l Conf. CyberWorlds*, 2009.

[19] O. Gallo, S. M. Arteaga and J. E. Davis, A camera-based pointing interface for mobile devices, *Proc. of IEEE ICIP*, 2008.

[20] J. P. Lewis, Fast normalized cross-correlation, *Vision Interface*, pp.120-123, 1995.

[21] D.-M. Tsai and C.-T. Lin, Fast normalized cross correlation for defect detection, *Pattern Recognition Letters*, vol.24, no.15, pp.2625-2631, 2003.

[22] I.-L. Jung and C.-S. Kim, Image enhancement using sorted histogram specification and POCS post-processing, *Proc. of IEEE ICIP*, pp.545-548, 2007.

[23] S. R. Subramanya and B. K. Yi, User interfaces for mobile content, *IEEE Computer*, pp.85-87, 2006.

[24] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 2nd Edition, Prentice Hall, 2002.

[25] D. G. Lowe, Object recognition from local scale-invariant features, *Proc. of IEEE CVPR*, pp.1150-1157, 1999.

[26] H. Bay, A. Ess, T. Tuytelaars and L. Van Gool, SURF: Speeded up robust features, *Computer Vision and Image Understanding*, vol.110, no.3, pp.346-359, 2008.

[27] T. Kailath, The divergence and Bhattacharyya distance measures in signal selection, *IEEE Trans. Comm. Technology*, vol.15, no.1, pp.52-60, 1967.

[28] M. Ebner, *Color Constancy*, The Wiley-IS&T Series in Imaging Science and Technology, 2007.

[29] S. Kesrovic, D. Mortenson and A. Nathan, An overview of managed/unmanaged code interoperability, *Microsoft Developer Network Library*, 2003.