Transforming a Regular Screen into a Touch Screen Using a Single Webcam

Longyu Zhang, Jamal Saboune, *Member, IEEE,* and Abdulmotaleb El Saddik, *Fellow, IEEE*

Abstract—Nowadays, touch screens are widely used in many commercial devices. In this paper, we present an inexpensive webcam-based touch screen system based on a simple and generic finger detection technique. Only a single webcam and a regular screen are used to achieve this goal. By using the scene geometry and the camera model, we can determine the position of the finger on the screen. The technique we propose is simple and works well under various surrounding environments while the detecting method suits many finger colors and shapes. Our approach can also be used for realtime applications. Experimental results and practical applications are provided to ascertain the proposed method.

Index Terms—Computer input-output, Computer interfaces

1 Introduction

In recent years, touch screens have been widely used in many commercial devices such as mobile phones, tablets, game machines and control panels. From an interaction point of view, touch screens have two main advantages: one is that they enable the user to interact directly with what is displayed rather than indirectly doing it with a pointer, which significantly improves the users' experience; the other is that they allow one to do so without requiring any intermediate devices that would need to be held in the hand. Given that, touch screens are used in situations where the keyboard and the mouse systems do not offer a suitably intuitive or rapid interaction. The only drawback of the currently commercialized touch screens is their high cost which is due to the usage of costly technologies (resistive or capacitive) [1].

A variety of touch screen technologies have been utilized to sense touch, and commonly used touch screens include resistive, capacitive, infrared-based, and webcam-based touch screens. Compared with other ones, webcam-based touch screen is less accurate, but the loss in accuracy is highly compensated by the low cost of the computer vision technologies used, which provides a big advantage over conventional touch screens. Also, it can easily transform most ordinary monitors into touch screens. Thus, we decided to adopt this method.

This paper describes a method to realize touch screen functions with a single webcam and a regular monitor, and thus it provides the potential to transfer touch screen applications onto inexpensive regular monitors, as well as ordinary laptop screens. Also, our system can be qualified as generic because it suits fingers with different colors and shapes, and does not need any learning or classification step. Moreover, the camera position we adopt makes the system not sensitive to changes in its environment. Furthermore, we developed a haptic video chat system and a writing and drawing application with our proposed touch screen to show that it is of practical use, and we will develop more applications in the future.

This paper is organized as follows. The second section introduces some related works about webcam-based touch screens and finger detection methods. In the third section, we discuss our proposed webcam-based touch screen method. The experimental results and developed practical applications are given in the fourth section. Finally we summarize our work and propose directions for future work.

2 Related work

A variety of approaches have been developed to realize touch screens. A resistive touchscreen panel consists of several layers, and among those layers, the two most important ones are thin, transparent electrically-resistive layers separated by a thin space. When an object or a finger presses down on the outer surface, the two layers touch and is connected at that point, and the touch position will be calculated from the obtained data [2]. A capacitive touchscreen panel consists of an insulator such as glass, coated with a transparent conductor. Since human body is an electrical conductor, touching the surface of the screen will lead to a distortion of the screen's electrostatic field, measurable as a change in capacitance [3]. An infrared touchscreen uses an array of X-Y infrared LED and photodetector pairs laid around the edges of the screen to detect a disruption in the pattern of LED beams. Xu [4] illustrates one kind of systems for detecting touch events by an optical touch screen where transmitters and receivers are positioned according to an alternating scheme on each side of the touchscreen.

- *L. Zhang, J. Saboune, and A. El Saddik are all with Multimedia Communications Research Laboratory (MCRLab), School of Electrical Engineering and Computer Science - University of Ottawa, Ottawa, Ontario, Canada*
- *E-mail: {lzhan121, jsaboune, elsaddik}@uottawa.ca*

Many webcam-based approaches have been proposed to overcome the high cost of the existing touch screen technologies. The principle of these methods is to detect the hand/finger in the video feed during the interaction and estimate its position when touching a regular screen, and as a result simulate a touch screen.

Cheng *et al.* [5] developed a novel 3D pointing system that allows users to interact with large displays by using a single webcam. With their system, users can use a bare hand to point naturally, and the virtual touch screen coincides with the large display, which makes the interaction more direct. Since a fingertip is hard to detect from a frontal view with segmentation techniques, users were asked to lower their fingertip to make sure that it would appear as the lowest skin color pixel. And users' body postures were restricted as slight deviation would increase the system's estimation error, which is mainly caused by the face detection inaccuracy.

Zhang *et al.* [6] realized their Visual Screen by putting a webcam along or not too far from a line normal to the center of the monitor and using the color difference between the hand color and the screen color to segment the human hand. Based on the detected hand, they were able to use finger features, such as finger tip shape, to find the finger tip location as the touch point. Visual Screen can also correct for the non-flatness of the computer screen to get accurate testing results. The disadvantage of this approach comes from the fact that the hand segmentation is simplistic and necessitates having an image of the screen background without the hand. Also, the hand detection will be erroneous if the screen displays colors similar to the flesh color. In addition, this method is very sensitive to any noise/object present in the camera field of view.

Zhou *et al.* [7] mounted two digital video cameras on the top left and right corners of an HDTV and installed two IR lights right beside each camera to develop a finger-friendly interactive board. Once a finger or pen touches the surface of the screen, it will cross the edge of the HDTV in the image, so they used this strategy to quickly find ROI (region of interesting). In our approach, we also adopt the similar strategy: active touch detection when a finger crosses the monitor's bottom border in the image. Their system has certain drawbacks. First, from their prototype, extra frames are required to fix the cameras, which makes it hard to set up. Second, IR lights are needed to make finger detection easily.

Hoang *et al.* [1] positioned one webcam (WC2) at the same location as that of the Visual Screen to detect the finger touch point, while using another additional webcam (WC1) to detect if the finger is really touching the screen. If a touch event is detected by WC1, it will activate WC2, and WC2 will begin detecting the finger tip position through three stages: preprocessing with color edge detection using color-space gradient, intermediate processing with motion-based detection, and pattern recognition using finger tip template matching. However, this method still suffers from the problem of occlusion and noise in the images. Also, it requires complex time consuming operations such as the fingers pattern recognition.

Realizing a webcam-based touch screen is mainly based on detecting the contact point between a finger and the screen using video feed from a webcam. Thus, detecting the finger tip in the image, which is always considered as the indicator of the touch position, is the foundation of this method. The first step in estimating the position of a finger tip on the screen using video is to detect and isolate the finger in the image. Many approaches have been developed to solve this problem. Most of these techniques rely on the shape or color of the hands and fingers.

Since human hands have distinct colors and shapes, many techniques used these features to segment the hands from the background, and then use these detected hands' contours to find the finger tips based on their special curvatures. Wang *et al.* [8] proves that the finger tips have high ratio curve in a hand contour after Fourier Transform, and usually their ratio curves are the highest when there is no deformation. This feature could be used for finger tips extraction.

Though many algorithms distinguish finger tips using the curvature features, they apply different methods to detect the hands. Some researchers use the original hand color ranges to find it. Kang *et al.* [9] assume that the skin color vary in certain ranges, and they use *OpenCV* to perform a faster RGB to YUV conversion in order to make the color ranges more robust to changes in brightness and intensity. They also minimize the effect of the palm by eliminating it from the image using morphological operations, before using contours to find the fingers. Lee *et al.* [10] use a webcam to detect a control motion first, and then zoom in the camera, when a control signal is given, to recognize the exact control gesture using the original skin color information, which is about relationships among finger's red, green and blue colors. Zhang *et al.* [6] observe in their experiments that the images of screen pixels have some degree of invariance in the color space even if contents change frequently, so they use the found dominating blue color to differentiate hand from the background. Though a hand may have certain color ranges, its colors could change according to the surrounding light conditions or the color reflection of nearby objects. Thus, Liang-Guo Zhang *et al.* [11] advanced a sign language recognition system based on wearable glove colors. Their left-hand glove is purple, while each finger of the right-hand glove has a specific color (red, yellow, orange, blue and green). First, they locate the reference point using the pupils detection. Then they use color difference to find hands and fingers. Since fingers of the right hand have different colors in this case, they can easily distinguish the position of each finger, and then calculate the exact finger postures including finger tips. As the glove colors are much more constant than natural finger colors, they are reliable features to detect. This method has a detection accuracy rate of 91%, but wearing gloves may affect the user's freedom. Given the uncertainty of the skin color under different lighting conditions, other features have been used to detect a hand in the image. The hand contour features can also be used to distuiguish hands

from other objects. Anagnostopoulos *et al.* [12] find all objects' contours in the image captured by a webcam, and use hand features like dimensions of the object area and its shape to wipe out non-hand objects. Then they trace the hand's contour curve, identify a set of dominant curve points and use k-curvature, which is mainly used to detect the dominant points on curves, to decide which points are finger tips. Since their system just distinguishes two types of real objects: hands and all the rest, their application supports multi-hands detection.

Since finger or hand shapes remain the same in different configuration or various scenes, some researchers also realized hand detection using pattern matching with pre-defined templates.

The Magic Crystal Ball [13] uses a finger tip template matching approach to verify the finger tip candidates which have passed the filtering step and remove the false matching candidates. In their process, candidates with low scores from the matching evaluation are discarded, and for false matching removal, if the pixels in the diagonal direction on the boundary of the finger tip patch coexist, then it is not considered as the finger tip and is removed; Hoang *et al.* [1] also use 60×80 templates with finger-like shapes as the specific patterns to match with sub-images in the images captured from a webcam. Then the sub-image with the highest correlation coefficient among all the sub-images is considered as the finger tip. Sometimes the templates learning database base may not contain templates representing all the possible configurations. Thus, an initialization and an extensive learning phase of the system is necessary for each user. Gorodnichy *et al.* [14] get the hand templates through ground subtraction in the initialization phase, and then use deformable templates to track the hand which could be highly deformable.

The methods exposed above rely heavily on complex image segmentation and pattern matching algorithms. Moreover, they are sensitive to illumination changes and occlusions. In order to avoid using complex pattern recognition techniques and color matching, we opted to take advantage of the camera positioning to find a generic and simple approach for finger detection. This technique will be explained in the next section. No learning nor database is needed as we do not use any shape or color matching.

Once the finger tip is detected, another challenge would be estimating its position. To achieve this goal, the most common technique would be to use a stereo-vision technique. Stereo-vision is mainly used to realize the 3D reconstruction using two or more cameras. From a general point of view, the 3D reconstruction problem is mathematically determined only if enough information about the sensing subsystem is available in terms of optical/intrinsic parameters and geometrical/extrinsic parameters [15]. With a 3D model of the finger, we can find its location in the image easily if no occlusion has occurred. In order to avoid these problems and the use of two cameras, we propose a new technique for position estimation.

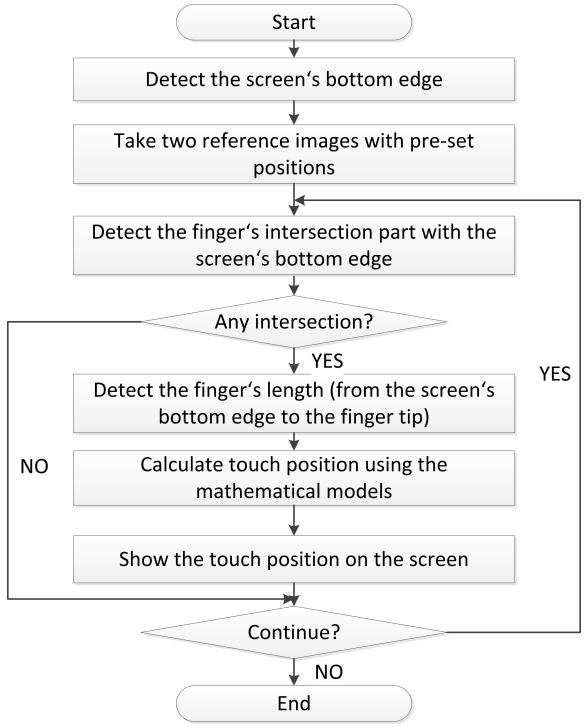


Fig. 1. Flowchart of our proposed method

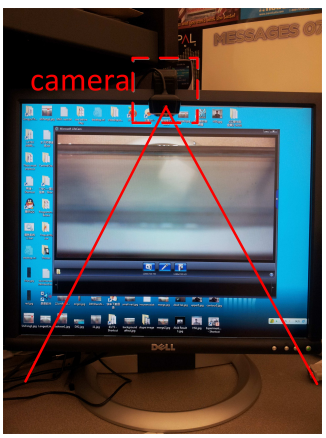


Fig. 2. Webcam position and its field of view

3 Proposed Webcam-Based Touch Screen Method

In this section, a detailed description of our proposed method is given. We start by introducing the webcam position we adopted and the advantages of this setting. Then we explain the approach we used to detect the finger in the image as well as the mathematical models enabling the estimation of the vertical and horizontal touch position. Details of the initialization and calibration phase are also exposed. Figure 1 shows the flowchart of our proposed method.

3.1 Webcam position

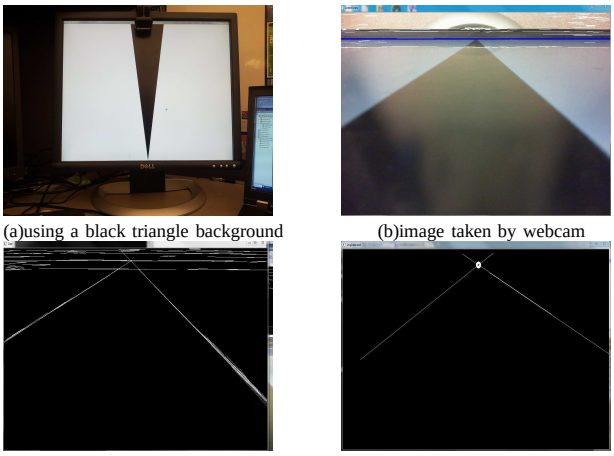
In our proposed method, we require a single webcam. In order to calculate the touch point between the finger and the monitor, we need to detect the finger from the image captured by the webcam. To make this complex task easier, we opt to position our webcam in a way to have the monitor's bottom edge horizontal in the image captured by the webcam. Given that configuration, the only object (in the image) intersecting the screen's bottom border during the finger-screen interaction would just be the finger itself. The finger detection in the image is then reduced to detecting the object intersecting the screen border. Figure 2 indicates the position and field of view of the webcam which is located on top of the monitor and looking down at the desk. Thus we can capture the finger length in the image and use it to calculate the touching position. From the view of the camera, we can see that only a little part of the desk is within the camera's field of view, while other surrounding environment objects will not be captured. As a result, the large ambient changes in the scene (noise, user movement, *etc*.) would not seriously affect the detection results or make the detection more difficult. Though the webcam's field of view covers only a triangle area instead of the whole screen, this webcam positioning makes the system less sensitive to surrounding environments and background changes, which significantly improves our system's robustness and precision. Moreover, our method has less restrictions on user's sitting position and posture, since most of the time the user is out of the view of the webcam.

3.2 Initialization and calibration phase

Our goal is to accomplish an accurate estimation of the contact point position between the finger and the screen, so an initialization and calibration phase is necessary for further calculation. During this phase, we will detect the reference line (the screen's bottom border) used for the finger detection, and collect data from two reference positions that are necessary for further calculations.

3.2.1 The screen's bottom border detection

As mentioned before, we active finger detection when a finger crosses the monitor's bottom border, so the first step in algorithm is finding the screen's bottom border. As a first step, we display a white image with a black triangle background (Figure 3(a)). The image taken by the camera will be Figure 3(b). We then use the *CVLINE* function provided by *OpenCV* to find all lines in the image taken by the webcam (Figure 3(c)). By adding conditions such as the lines' lengths and inclination angles, we can easily find the black triangle's two edges from all the detected lines, then we can calculate the intersecting point of those two lines, as shown in Figure 3(d), the white solid circle is the point of intersection. Since we assume that the monitor's bottom edge is horizontal in the image captured by the camera, the horizontal line passing by this white dot will be the screen's bottom border (the blue line in Figure 3(b)). The



(a)using a black triangle background (b)image taken by webcam

(c)lines found from the image (d)the detected intersecting point

Fig. 3. Screen's bottom border detection

reason we utilize triangle background is that screen color near the monitor border is usually affected by the reflection of the border from the view of the camera, using triangle edges, which can be more easily and accurately detected, can help us locate the triangle vertex (a point on the border) even if there is no large color difference there.

3.2.2 Two reference images with pre-set positions

As we are aiming to develop a generic system that can be adapted to any webcam and situation, the estimation of the used camera's position and focal length is necessary. This estimation is based on a simple camera model (no distortion or optical center displacement are taken into account).

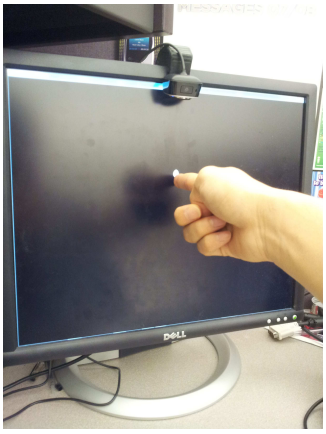


Fig. 4. User's finger is pointing the first indicated position (the white dots)

In this step, we try to acquire some reference images with the user's finger pointing to two well known positions in the screen separately. In our system, one pre-set position is the screen's horizontal midpoint at 10cm deep from the screen's top border when the screen's height is 30cm; the other position is 101 pixels to the right of the screen's horizontal midpoint (screen resolution is 1280×1024) at 22cm deep from the top border. For each step, a white dot

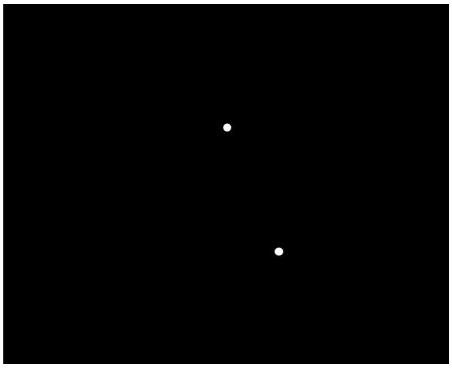


Fig. 5. The two pre-set positions (white dots)

shows the indicated pointing position in a full-screen black window. Actually, any two dots on both different vertical and horizontal lines can be selected as the two pre-set positions, but certain recognizable distance between these two positions makes the system more robust. Then the user can point and touch the white dot while keeping his/her finger normal to the screen, as shown in Figure 4. Figure 5 shows the two pre-set positions. In the real system, these two dots will not be shown at the same time, we just put them together to illustrate our technique.

With those steps, we have finished the initialization and calibration phase, and the images and data collected from those steps will be used in the following sections. In our design, users do not need to press the monitor hard, slightly touch will be fine. If touch causes vibration of the webcam, then recalibration is required.

3.3 Locating the fingertip in the camera image

Since we consider finger itself as a whole part has similar color, we first locate the finger's intersecting part with the monitor's bottom border from the view of the webcam, and then use this part's color as the standard to further find the whole finger. Compared with methods using constant color range, our approach suits more fingers with different colors.

3.3.1 Finding the intersecting part of the finger

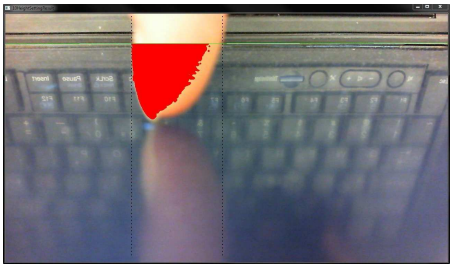


Fig. 6. Finger detection result at 10cm (from view of webcam)

When the finger touches the screen, it will be the only object intersecting the detected bottom border line from the camera's field of view, as shown in Figure 6. Since

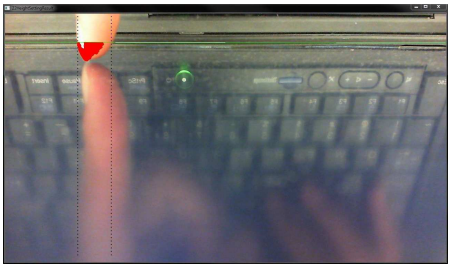


Fig. 7. Finger detection result at 22cm

the red components of the finger and the monitor's bottom edge have large difference, we extract the red color value from the RGB color components of the screen's bottom border line and compare the red color of all pixels on this line, then the two locations with the largest color difference will be the finger edges. To save time when searching, we use a pseudo gradient descent technique. First, we divide the line into several large groups (*e.g.* 10 points each group), and calculate each group's average red value. Then compare every two adjacent groups' value, the two pairs of groups with the biggest difference values are considered as the initial edges separating the screen and the finger. A more accurate positioning of these lines is then accomplished by applying the same technique on the 30 points interval surrounding the initial edges. In fact, we divide these intervals into smaller groups of 3 points and thus can obtain the exact finger's position. The two dotted vertical lines in Figure 6 and Figure 7 show the detected finger edges separately.

To avoid some small isolated dots, which may be present on the finger or the monitor's bottom edge, and might cause sudden color change, we process the image with erosion, dilation, and smoothing functions before any detection. Also, the distance between the two found lines should not be too short as finger in the image should not be very small.

3.3.2 Determining which pixels belong to the finger

After finding the exact finger edges from the screen's bottom edge, we can easily locate the central part of the finger at this line (from 1/4 to 3/4 of the finger's width). Then we extract these central part pixels' color components and calculate their average value; these color values will be used as the standard to find all qualified pixels, whose color components' differences with the standard value are within certain thresholds (RGB color components' possible values are from 0 to 255, while Hue color component varies from 0 to 179), and are located between the two vertical blue lines (Figure 6 and 7). At first, we were using only RGB color model for detection, but later we found that the RGB model is too sensitive to light reflection, especially when the finger is so close to the screen in our case. Thus, we also use the HSV color space for the detection. The HSV stands for the Hue, Saturation and Value. The HSV components correspond more closely to human-perceived qualities of a colour than the RGB components [16], so it is widely used for human hand or face detection. However, HSV values

are also affected by screen colors sometimes, so we came up with the idea of using both HSV and RGB models, and it works well. For each color component, we use different color thresholds to determine similar colors when detecting the whole finger. After lots of tests, using the following thresholds gave the best result: the red threshold is ± 50, the blue threshold is ± 70, the green threshold is ± 70, while the hue threshold is relatively low (± 25). Lowering the thresholds will just detect part of the finger, when raising it may mistake non-finger pixels as the finger. Pixels with all these four qualified color components and located between two detected finger lines will be considered as finger pixels and are shown in red (other colors are also fine) in Figures 6 to 9. The red parts in Figure 6 and Figure 7 show the finger detection result at 10cm and 22cm deep separately.

The touch point would be considered as the midpoint of the lowest level containing the last qualified group of pixels in the image (either fingertip or finger nail depending on the user's finger shape). Though some parts of the finger are not detected in Figure 6 due to the light reflection and the finger's intrinsic color difference, they do not affect our detection accuracy in this case, since in our mathematical models, only the detected finger length (vertical direction) is used, while the finger width (horizontal direction) is mainly used to find more accurate finger length.

3.3.3 More filtering for finger detection

To obtain better finger detection results, we also adopt some strategies to improve the accuracy of our detection algorithm. In our method, when we search for the whole finger between the two detected finger edge lines, using only the color threshold we may falsely make "non-finger" pixels qualified as finger, such as the mirrored finger pixels produced by reflection on the screen (Figure 8) or the pixels with similar colors in the background (Figure 9). In order to solve this problem, we add two filtering steps to our finger detection algorithm.

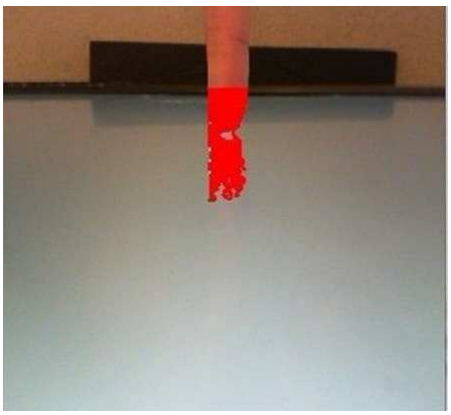


Fig. 8. Some pixels representing the mirror finger are mistaken as belonging to the real finger given their color features

One filtering is the continuity property. Since a finger is a single connected object, we exploit this fact when identifying finger pixels by searching downwards from the bottom border of the screen and stopping when we reach a line whose finger-colored pixels (if any) are not positioned near enough to corresponding finger-colored pixels on the previous row of the image. To explain our filtering better, we rank the screen's bottom border as the 1st line, then horizontal lines (each horizontal line is a line of pixels in the image) below it will be ranked as 2nd, 3rd... line in order. After we find the intersection part between the finger and the border, for the first 3 horizontal lines we just use the color conditions mentioned before to select the qualified pixels between the two detected finger edge lines. From the 4th line downwards, only when more than 2/3 of the pixels located between the leftmost and the rightmost pixels which satisfy our color requirements on a given line, are directly below some qualified pixels on the previous line (such as qualified pixels on the 3rd line for the 4th line detection), we consider the qualified pixels on this line as finger parts; otherwise, we will consider this line as the finger tip and stop searching. Thus, the problem is solved by adding this additional continuity requirement. We set this requirement as we consider that the finger is a plain object with a uniform color. Thus, all pixels belonging to it should have similar color and should occupy most of the pixels between its borders.

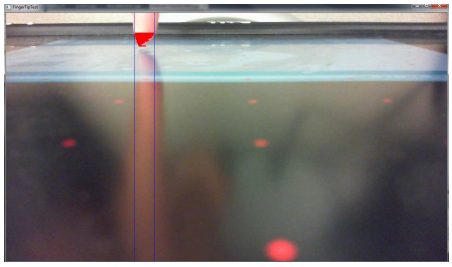


Fig. 9. Our strategy works well to stop the large increase in finger width

The second filtering is the shape property. Despite the first filtering step, some non-finger pixels (such as the mirrored finger pixels) are close enough in color to the real finger pixels that they all pass the first filter. Thus, we need to add another step based on the finger's shape to further increase the accuracy of our detection result.

Given the shape of the finger, when going from the 1st horizontal line to the finger tip, the finger width should not increase greatly in most cases, and actually the width would decrease gradually. Thus, we also observe the finger width changes when detecting the whole finger to rule out non-finger elements. When scanning the "finger" from the 1st line to the bottom of the image with color and continuity condition, we disqualify all the pixels belonging to any horizontal level with sudden large finger width increase. Those disqualified pixels may represent the mirror finger on the screen or pixels with similar colors in the background. Figure 9 shows that the finger detection scanning process is stopped when a large increase of the finger width happens below finger tip. In this situation, the touch point would be

considered as the midpoint of the level containing the last qualified group of pixels.

Using this strategy, we successfully eliminate some false detections based on the finger's shape feature.

3.4 Realtime touch position calculation

The realtime finger detection uses the same finger detection method mentioned in the initialization and calibration phase, and the color difference thresholds we adopt are still the same. After finishing detecting the finger in a image, we can get the current finger length l in the image, and then calculate the finger's touching point's horizontal position F_h and its vertical position F_v on the monitor with our mathematical models. Once we get F_h and F_v, we have acquired the touch position. As a result, we realize a touch screen with a webcam. Our mathematical models are based on the camera pinhole model.

3.4.1 Realtime finger's vertical position calculation

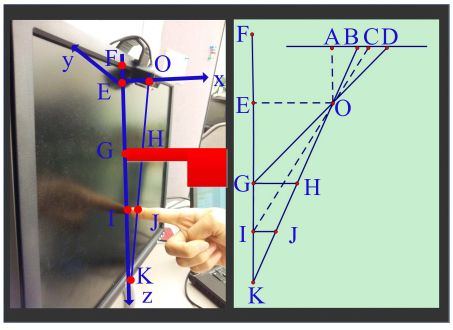


Fig. 10. Combination of 10cm and 22cm calibrations and the mathematical model

The finger's vertical position (depth to the screen's top border) calculation is mainly based on the 10cm and 22cm reference images and the pinhole model of the camera. Actually, given any two configurations of the finger touching the screen at any two points at different heights, we can establish the following mathematical model (equation 1 to 7). Figure 10 shows the picture of these two calibrations' combination, and indicates the mathematical model we adopt. Though the camera, the 10cm and 22cm calibrations are not on the same plane, we project them on the same plane to build the model, and neither the finger length nor any vertical distance will change. In fact, in the real application we are not going to touch those two points at the same time; we combine them together in one image just for explaining the mathematical model better. In this figure, we assume that the red virtual hand is pointing at a point 10cm deep from the top edge of the screen, and the real hand is pointing at a point 22cm deep from the top edge of the screen.

In the mathematical model,

- The point O refers to the optical centre of the camera, and point E is point O's perpendicular projection on the screen. On the screen, we draw a line through E and we make it perpendicular to the webcam's image plane. Since we assume the monitor's bottom edge is horizontal in the captured image, the drawn line will also be perpendicular to the screen's top and bottom border at the same time. F marks the intersection point of this line and the screen's top border, while K is the point on screen's bottom edge, so FK refers to the monitor's vertical line whose length is equal to the screen height m_h (30cm in our case). EF is then the vertical distance d between the camera's optical centre and the screen's top edge.;
- The line AD represents a line on the camera's image plane, which is used to form images captured through the optical centre, and OA is the vertical distance between the camera's image plane and the camera's optical centre;
- G and I are the finger's touching positions on the screen. If we assume they are the 10cm and 22cm calibration settings, we can get that FG=10cm (h_1) and FI=22cm (h_2). Actually, if the webcam is not located right on the middle point of the screen, we can project the touch positions to this line, and the detection results will not change;
- OK will then be the line joining the optical centre O to the screen's bottom border point K, and thus will intersect the finger in different positions (H, J) depending on their distance (depth) to the camera. Then GH and IJ refer to the 3D portions (of lengths l'_1 l'_2) of the detected touching finger length, and they are represented in the camera's image plane by BD and BC (l_1 and l_2).

Given the webcam position, we assume that the line OA is perpendicular to both AD and EO, and EK is perpendicular to EO. Thus, from the similar triangles $\triangle OGH$ and $\triangle ODB$, we can state that

$$GH/DB = (FG - EF)/OA \tag{1}$$

Same for the similar triangles $\triangle OIJ$ and $\triangle OCB$

$$IJ/CB = (FI - EF)/OA \tag{2}$$

The relationship between IJ and GH can be derived from similar triangles $\triangle KIJ$ and $\triangle KGH$

$$IJ/GH = (KF - FI)/(KF - FG) \tag{3}$$

If we stop considering IF as the 22cm deep calibration setting, but assume that it is a random height, namely the finger's vertical position F_v (IF) is unknown, we can use equation (1),(2),(3), and BC which then refers to the detected finger length l (pixels) in image, to calculate IF

$$IF = \frac{BD * FK * (FG - EF) + EF * BC * (FK - FG)}{BC * FK - BC * FG - BD * EF + BD * FG} \tag{4}$$

namely

$$F_v = \frac{l_1 * m_h * (h_1 - d) + d * l * (m_h - h_1)}{l * m_h - l * h_1 - l_1 * d + l_1 * h_1} \quad (5)$$

In equation (4) and (5), the only value that remains unknown is EF (d), the vertical distance between the screen's top border and webcam's optical centre. So we need to use both the 10cm and 22cm reference images to calculate EF, in other words, when FG=10cm, IF=22cm, and BC is l_2. From equation (1),(2) and (3), we can get

$$EF = \frac{FK*FI*BC - FG*FI*BC - FK*FG*BD + FG*FI*BD}{FI*BD + FK*BC - FK*BD - FG*BC} \quad (6)$$

namely

$$d = \frac{m_h * h_2 * l_2 - h_1 * h_2 * l_2 - m_h * h_1 * l_1 + h_1 * h_2 * l_1}{h_2 * l_1 + m_h * l_2 - m_h * l_1 - h_1 * l_2} \quad (7)$$

Thus, once we have the distance EF(d), then in realtime detection we just need data from one of the two reference images to calculate any realtime finger touch's vertical position with equation (4) or (5).

In fact, if the camera's optical centre is higher than the top border of the screen,this mathematical model will still work well.

3.4.2 Realtime finger's horizontal position calculation

The finger's horizontal position calculation requires information about the two calibration positions, the monitor's resolution, the calculated distance d from the camera's optical centre to the monitor's top border, and the calculated finger's vertical position F_v. With those data, we will be able to calculate the touch point's horizontal position. This calculation is done as follows.

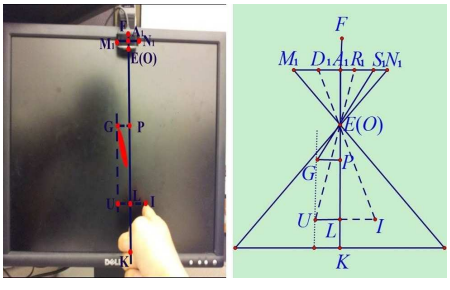


Fig. 11. Mathematical model for horizontal position calculation

Figure 11 shows both the real scene and our proposed mathematical model for horizontal touch position calculation. In this model,

- Same as the mathematical model we use in Figure 10, F is a point on the screen's top border, K is a point of screen's bottom edge, and FK refers to the monitor's vertical line whose length is equal to the screen height m_h (30cm in our case). The optical centre O's perpendicular projection on the screen is E. Thus EF has the same value we get from equation (6);
- The line M_1N_1 refers to the camera's image plane's horizontal projection line on the screen (this line M_1N_1 is perpendicular to the image plane's line AD in Figure 10). A_1 is point E's perpendicular projection on line M_1N_1. The reason we do this projection is to put all related points and lines on the same plane (the screen plane) for an easier mathematical model building, and this projection will not affect the object's width and horizontal position in the camera's horizontal image plane;
- The line GU is the vertical middle line of the screen, and the Point G is the first reference touch point, which is the screen's horizontal midpoint at 10cm deep from the screen's top border. I is the second reference point, which is 101 pixels to the right of the screen's horizontal midpoint at 22cm deep from the top border. Line EK is the line passing the camera's optical centre's projection position E, and is perpendicular to M_1N_1. L and U are I's projection position on line EK and GU respectively, and P is G's projection position on line EK.

In our model, we assume the monitor's bottom edge is horizontal in the captured image, which means GU is parallel to EK. If the optical centre of the camera is right in the middle of the screen, GU and EK will be on the same line. However, we cannot guarantee this, so each time we calculate U's position in the image plane R_1 from the reference point G's position in the image plane S_1.

For similar triangles $\triangle EGP$ and $\triangle ES_1A_1$, we know

$$GP/S_1A_1 = (FP - EF)/EA_1 \quad (8)$$

Same for similar triangles $\triangle EUL$ and $\triangle ER_1A_1$

$$UL/R_1A_1 = (FL - EF)/EA_1 \quad (9)$$

Since GU is parallel to EK, we can get

$$GP = UL \quad (10)$$

From equation (8),(9), and (10), we can get

$$R_1A_1 = \frac{FP - EF}{FL - EF} * S_1A_1 \quad (11)$$

In this equation, if I is a random touch point at height F_v (which has been calculated from former step), we can calculate this height's screen's midpoint's position in the camera's image plane.

Then we can also get I's position D_1 in the camera's image plane, and its distance from R_1. From similar triangles $\triangle EUI$ and $\triangle ER_1D_1$, we know

$$UI/R_1D_1 = (FL - EF)/EA_1 \quad (12)$$

From equation (12),

$$EA_1 = R_1D_1 * (FL - EF)/UI \quad (13)$$

From equation (13), we can calculate the value of EA_1.

If we assume I is a random touch point instead of the second reference touch point, then UI will be unknown, but we can still get its height F_v from the former step, and its height's screen midpoint's position R_1 in the camera's image plane from equation (11). To calculate UI, we need equation (12) and the value of EA_1.

$$UI = R_1D_1 * (FL - EF)/EA_1 \tag{14}$$

Knowing UI, we have got the touch point's horizontal position F_h on the screen.

Same as for the mathematical model of vertical position calculation, this mathematical model also works when the camera's optical centre is higher than the top border of the screen

With the vertical and horizontal positions (F_v and F_h) of the touch position, and the screen's resolution, we can transfer the touch point's position onto the screen precisely, and could react with corresponding actions. So far, we have successfully changed an ordinary monitor into an touch screen with a webcam.

4 Experimental Results

Experiments are conducted to test the performance of our proposed system, and this section mainly gives details about the implementations, as well as the testing results of our touch screen approach with several backgrounds or the webcam directly, which have various resolutions, diverse backgrounds, and/or different users with varied clothes. Besides, we tested our system in different locations of our lab and some meeting rooms, and results show no substantial difference under these normal lighting conditions. Since our touch screen can work with realtime applications, we also successfully realized the application of a haptic video chat system and a writing and drawing system with it.

4.1 Hardware and software information

The implementation of our proposed touch screen is conducted in our laboratory with the following hardware and software.

- PC configuration: The PC we use has Intel(R) Core(TM) i7-2760QM CPU 2.4GHz with Window 7 64-bit Operating System, and its memory is 8GB.
- Monitors: The proposed methods are tested on both desktop and laptop monitors: one is a Dell 19.3-inch 1905FP monitor, and the other is a Lenovo ThinkPad W520 laptop screen. The desktop monitor's resolution is 1280×1024, and the laptop's is 1600×900, and their lighting conditions are slightly different.
- Webcam: We choose the Microsoft LifeCam HD-5001 [17] as our webcam to capture images. We select this webcam because its structure makes it easy to be positioned on top of the monitor and this webcam offers up to 30 high-resolution frames (1280×720 pixels) per second. Moreover, it allows setting the camera parameters, such as focus, brightness, white balance, saturation, exposure and contrast. The auto-focus function is disabled in the test in case of catching blur images during its focus process. Following are the used values: focus is 11, brightness is 133, contrast is 5, saturation is 83, exposure is -15, and white balance setting is automatic.
- Programming Environment: Our algorithm is implemented using Microsoft Visual Studio 2010 C++ with OpenCV (Open Source Computer Vision) [18] which is a library of programming functions for realtime computer vision, and the OpenCV version we adopt is 2.3.1.

4.2 Experimental result of the touchscreen

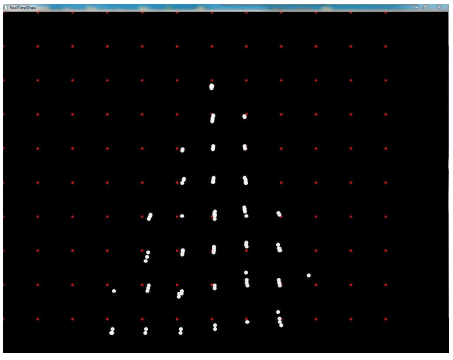


Fig. 12. Full-screen black window with red test points and white calculated points

To test the precision of our webcam-based touch screen, we pre-set several red test points on a full-screen window, and ask users to touch those dots within the webcam detection region. By calculating horizontal and vertical touch points' position on the screen, we record those points and draw them on the same window with white dots. Figure 12 is an example with pure black background.

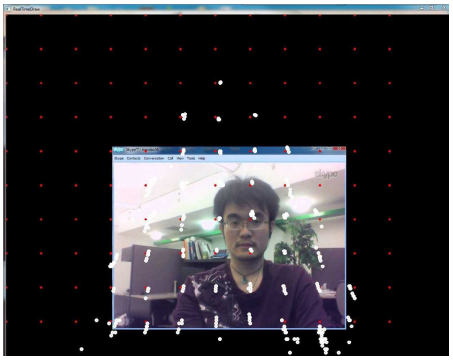


Fig. 13. Testing result with a *Skype* chatting image as background

We test our method with different backgrounds as the colors of the background images might affect the finger detection result. As one of the intended applications of this approach is to detect touch positions when chatting,

(a) Original figures (b) Figures with test results

Fig. 14. Testing results with other chatting images as backgrounds

we also capture several chatting images from *Skype* or webcam, and take them as part of the window background separately to test the compatibility between our application and the chatting software. In order to test as many situations as possible, the resolutions of these chatting images differ from 1280×853 to 640×480; lighting conditions and backgrounds vary with chatting environment, while colors of clothes are also different. Figure 13 shows a full-screen window with a *Skype* chatting image and our experimental results (white dots). Figure 14 shows other original images (left column) and our test results with those images (right column). Those images are tested on the desktop monitor. Detection failures (errors larger than 2cm) mainly happen at the low part of the screen, because only a small part of the finger will be shown in the image and a little error in finger detection could cause large touch position difference at this area. This is also why many white dots appear at the bottom of the screen.

In order to show more details of our experimental results, we add the boxplots of the touch points' vertical and horizontal absolute errors with these 6 images, as shown in Figure 15. All errors larger than 2cm are considered as detection failures and treated as a large value (3cm for both horizontal and vertical errors) when estimating the test output descriptors (e.g when calculating average error). From these experimental results, we can see that though there are some errors, most calculated positions are very close to the real touch positions. In the test, light from the background image may be reflected on the fingertip

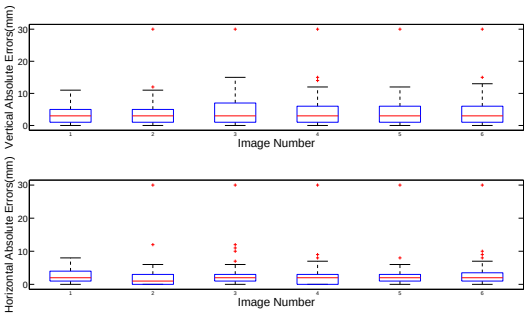


Fig. 15. The boxplots of touch points' vertical and horizontal absolute errors

and affect its color and further influence the detection and calculation result. Usually, white color would bring the greatest impact since its color reflection is the strongest among all colors, and it does have the similar color as real finger sometimes.

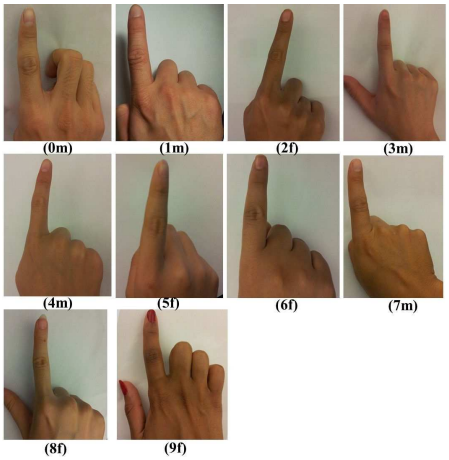


Fig. 16. Hands of all testers

Also, we test our touch screen with more users on the laptop screen. Testers 0, 1, 3, 4 and 7 are male, others are female. Figure 16 shows the hands of all testers. As we can see, testers' finger colors are various, and we have both long and short nails, as well as red nail polish and natural nail, which make the experimental results more reliable.

Furthermore, we evaluate the accuracy of our touch screen by calculating errors median, 75th and 90th percentile, worst-case errors, and the average value of calculated points (white dots) within each image. For these testing images, vertical and horizontal distances between two close pre-set red dots are 3cm on desktop monitor and 2cm on laptop screen. For convenience of managing images, we number the picture with black background in Figure 12 as Image 1, picture with a *Skype* chatting window in Figure 13 as Image 2, and pictures in right column of Figure 14 as Image 3 to 6. Results are shown in Table 1. Since occasional finger detection failures (errors larger than 2cm) occur for some points which are close to the

TABLE 1
Accuracy of Calculated Touch Positions

Image Num.	User Num.	Num. of Test Points	Num of Detection Failure	Detection Failure Ratio(%)	Vertical Abs Errors in mm					Horizontal Abs Errors in mm				
					Median	75th Percentile	90th Percentile	Worst Case	Avg.	Median	75th Percentile	90th Percentile	Worst Case	Avg.
1	0	74	0	0	3	5	7.4	11	3.46	2	4	6	8	2.61
2	0	166	7	4.22	3	5	10	25	4.40	1	3	5	24	3.02
3	0	142	3	2.11	3	7	10	23	4.70	2	3	4	12	2.65
4	0	148	7	4.73	3	6	12	25	4.94	2	3	6	15	3.35
5	0	151	14	9.27	3	6	9	25	5.73	2	3	6	18	4.51
6	0	104	1	0.96	3	6	10	27	4.36	2	3.25	5	10	2.60
1	1	67	0	0	3	4.5	6.4	9	3.07	1	3	4	9	1.78
2	1	60	0	0	3	5	7	10	3.50	2	3	3.1	5	1.72
1	2	44	2	4.55	2	3	4	30	3.02	4	6	9.4	17	5.36
2	2	33	2	6.06	2	4	5	10	3.73	3	8	10.8	20	5.58
1	3	51	5	9.80	4	5.5	10	15	6.10	4	6.5	12	23	6.75
2	3	37	0	0	2	4	5.4	14	2.97	2	6	10.4	14	3.81
1	4	50	4	8	5	7	10.1	22	6.72	3	5	11	11	5.06
2	4	38	0	0	2	4	4.3	7	2.37	2	5.75	6.3	10	2.89
1	5	45	7	15.56	2	7	F	32	7.11	5	9	F	12	8.18
2	5	35	6	17.14	2	7	F	22	7.26	5	6.5	F	15	7.89
1	6	48	1	2.08	3	4	6	30	3.58	3	5	6	10	3.56
2	6	51	2	3.92	2	4	6	22	3.47	1	2.5	3	17	2.80
1	7	47	4	8.51	5	8	10	23	7.17	3	5	7.4	8	5.00
2	7	39	0	0	3	5	6	6	3.08	3	5	6.2	9	3.18
1	8	56	10	17.86	6	8	F	30	9.09	2	4	F	16	6.82
2	8	50	3	6	2	5	6	22	4.20	1	2	4	5	2.92
1	9	59	7	11.86	8	11.5	F	32	10.08	4	7	F	13	7.03
2	9	54	4	7.41	4.5	7	11.7	32	6.33	2	4	6	10	4.35
Avg.		68.71	3.71	5.84	3.27	5.77	7.82	21	4.97	2.54	4.69	6.58	13	4.18

bottom line of the screen, we also show them in the table. The numbers of test points in the table are various as the results of non-fixed webcam's field of view and the fact that some users maintained their fingers longer on the dots and thus their pointing positions were considered twice or more as different dots/test points. However, the number difference should not cause substantial influence when we evaluate the results. From this table, we can see that the total detection failure ratio is 5.84%, and vertical and horizontal absolute errors medians are 3.27(mm) and 2.54(mm) separately which are rather small compared with the whole screen's height (desktop/laptop 300mm/195mm) and width (desktop/laptop 380mm/345mm). Also, for each image, the detection and calculation time is from 0.1s to 1.3s with the frame size 1280×1024. The closer the finger is to the camera, the longer the time will be, because more pixels need to be processed. However, if we disable the continuity property strategy which costs much time, the time will be reduced to 0.1s to 0.3s, but the system's accuracy will degrade a lot. We are still working on shortening the time by optimizing the software, using GPU and improving the strategy to suit more applications.

4.3 Applications

To prove the practical use of our system, we also developed some realtime applications with it.

4.3.1 A haptic video chat system

The original motivation for this touch screen is to combine it with teleconferencing software to develop a haptic video chat system. Our idea is to map the touched body part in the image to a corresponding vibrators to simulate a touch sensation at this part. This is realized with the following approach. As Figure 17 shows, first we utilize Haar Classifier function provided by *OpenCV* to detect the human face in a chat image. Then we estimate the shoulder width based on the detected face. Vitruvian Man [19] (sometimes called the Canon of Proportions), a world-renowned drawing created by Leonardo da Vinci, is based on the correlations of ideal human proportions along with the geometry described by ancient Roman architect Vitruvius. According to Leonardo's lower section of text in the accompany text with this drawing: "from the hairline to the bottom of the chin is one-tenth of the height of a man" and "the maximum width of the shoulders is a quarter of the height of a man". If we consider the face length we detect

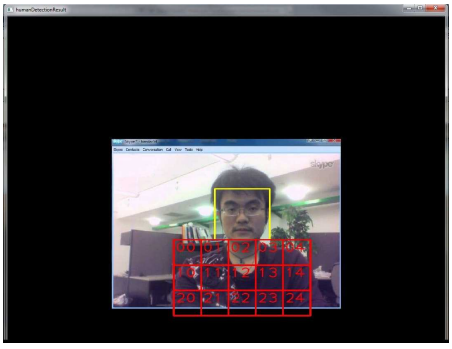


Fig. 17. Face detection and upper body dividing result

as the distance from hairline to the bottom of the chin, we can estimate that the maximum width of the shoulders is 2.5 times this face length. We then divide the shoulder width into 5 parts, and use squares to number those parts. In Figure 17, face detection result is shown with yellow box, while upper body dividing results are displayed with red boxes and numbers. As we can see, this kind of upper body detection and division method could offer a good estimation result though it is not extremely accurate, but this level of estimation would be enough to satisfy our request of transmitting emotion through a touch screen.

Fig. 18. Touch activates number 08 box

After finishing the human body detection, once a touch is detected in a specific square (box), as shown in Figure 18, our system will show the touch region number ("08"), and inform the control centre of a haptic jacket to activate the corresponding vibrators to produce the sense of touch. Since the passive user is wearing a haptic jacket, he/she will feel this touch. From the feedback of 15 users, this application received score 4.25 (out of 5) for its accuracy, 4 for its realistic haptic feedback, and 4.25 for the acceptance of its delay and response time.

4.3.2 Writing and drawing system

Besides using the touch screen for telecommunications, we also use it in educational or recreational applications, such as drawing, writing, and games. By adding control keys like "stop", "start", and "clear background", we have used our touch screen to realize functions of drawing and writing. Figure 19 shows the results of writing the word "MCR". Figure 20 shows the drawing of a human figure. Of course, we can also realize more applications with our proposed systems, and we would try to do it in the future.

5 Evaluation of the System

Though our system can work with some realtime applications, we can still improve it in the following fields.

First, as shown in Figure 2, the view of the webcam covers only the triangle area under the two red lines instead of the whole screen. Raising position of the webcam is able to solve this problem, but extra frames will be required to fix the webcam, which is against our idea of proposing a simple and generic system.

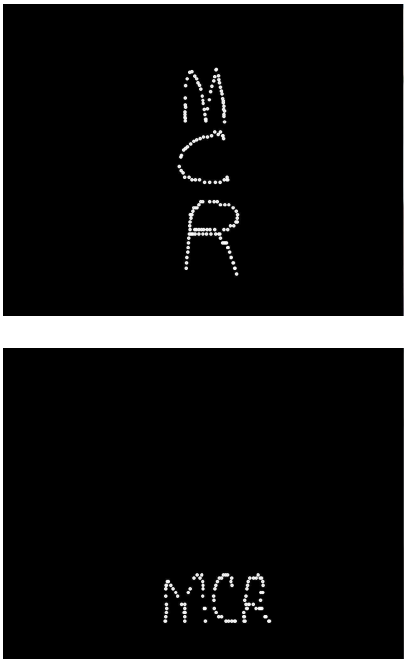


Fig. 19. Writing word "MCR"

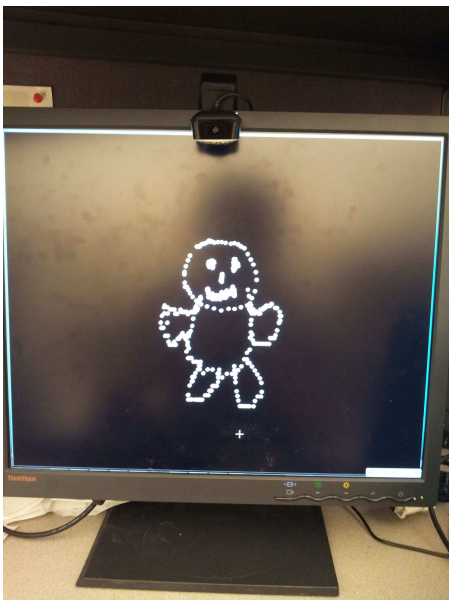


Fig. 20. Figure drawing

Second, currently our approach works only with a single finger, supporting multi-touch will make it more interesting, but a multi-finger detection method is required. A main problem is that as we use just a single webcam, if fingers overlap from the field view of the webcam, then actions will not be detected. Besides, Since our method detects the finger's position using its uniform color properties, it can be used to detect and estimate the position of any object of a uniform single color (e.g stylus, pen). This property gives our approach an edge over capacitive touch screen as it allows the usage of any object for interaction and not only fingers (or capacitive material). In fact, more tests with

different types of objects will be conducted soon to assess the validity of this approach.

Third, compared with touchscreen cellphones and tablets, the system's accuracy and process time can still be improved, as well as minimizing restrictions for finger poses, but its low cost highly compensates the drawbacks, and the ability to transform tradition screens also increases its competitive power.

Despite the limitations mentioned above, our proposed system is still in practical use with applications which requires just a part of the monitors, acceptable update rate and accuracy. The haptic video chat system is apparently a good example of these kinds of applications that fit the touchscreen system well.

6 Conclusion and Future Work

In this paper, we have presented a touch screen system with a single webcam and a regular monitor, and developed some practical applications like haptic video chat system and writing and drawing system with this touch screen. Our system works well with different backgrounds due to the camera setting. Moreover, the finger detection method we adopt suits many finger colors or shapes, and do not need any learning or classification step. The experimental results show that this system can also be used in realtime applications. Also, our method is able to transform most surfaces into a touch screen.

Related videos can be found on *Youtube* website [20], [21], [22].

This touch screen system is still under research, and the further work will focus on improving the system's precision and finger detection success rate, as well as optimizing the calibration phase. Without continuity strategy, we observed from the preliminary tests that the detection accuracy was lower but no formal study was conducted to evaluate this deterioration. Thus, we also plan to acquire quantitative results about how accuracy degrades when "continuity strategy" is disabled, and try to test if geometry correction can be added to our system.

References

[1] M. P. Hoang, T. D. Nguyen, and T. M. Hoang, "Implementation of webcam-based touchscreen," in *Communications and Electronics (ICCE), 2010 Third International Conference on*, aug. 2010, pp. 201 –206.

[2] K. T. Robert Phares, "Resistive touchscreen having multiple selectable regions for pressure discrimination," Patent, 09 1998, uS 5815141. [Online]. Available: http://www.patentlens.net/patentlens/patent/US_5815141/en/

[3] D. O. Robert G. Kable and C. O. Philip A. Schlosser, "Electrographic apparatus," Patent, 05 1987, uS 4665283. [Online]. Available: http://www.patentlens.net/patentlens/patent/US_4665283/en/

[4] L. XU, "Infrared touch screen," Patent Application, 05 2011, uS 2011/0115748 A1. [Online]. Available: http://www.patentlens.net/patentlens/patent/US_2011_0115748_A1/en/

[5] K. Cheng and M. Takatsuka, "Initial evaluation of a bare-hand interaction technique for large displays using a webcam," in *Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems*, ser. EICS '09. New York, NY, USA: ACM, 2009, pp. 291–296.

[6] Z. Zhang and Y. Shan, "Visual screen: Transforming an ordinary screen into a touch screen," in *Proceedings of IAPR Workshop on Machine Vision Applications*, 2000, pp. 215–218.

[7] Y. Zhou and G. Morrison, "A real-time algorithm for finger detection in a camera based finger-friendly interactive board system," in *The 5th International Conference on Computer Vision Systems*, Bielefeld, March 2007, pp. 21. – 24.

[8] X. Wang, X. Zhang, and G. Dai, "Tracking of deformable human hand in real time as continuous input for gesture-based interaction," in *Proceedings of the 12th international conference on Intelligent user interfaces*, ser. IUI '07. New York, NY, USA: ACM, 2007, pp. 235–242.

[9] S. K. Kang, M. Y. Nam, and P. K. Rhee, "Color based hand and finger detection technology for user interaction," in *Convergence and Hybrid Information Technology, 2008. ICHIT '08. International Conference on*, aug. 2008, pp. 229 –236.

[10] D. Lee and Y. Park, "Vision-based remote control system by motion detection and open finger counting," *Consumer Electronics, IEEE Transactions on*, vol. 55, no. 4, pp. 2308–2313, 2009.

[11] L.-G. Zhang, Y. Chen, G. Fang, X. Chen, and W. Gao, "A vision-based sign language recognition system using tied-mixture density hmm," in *Proceedings of the 6th international conference on Multimodal interfaces*, ser. ICMI '04. New York, NY, USA: ACM, 2004, pp. 198–204.

[12] A. Anagnostopoulos and A. Pnevmatikakis, "A realtime mixed reality system for seamless interaction between real and virtual objects," in *Proceedings of the 3rd international conference on Digital Interactive Media in Entertainment and Arts*, ser. DIMEA '08. New York, NY, USA: ACM, 2008, pp. 199–204.

[13] L.-W. Chan, Y.-F. Chuang, M.-C. Yu, Y.-L. Chao, M.-S. Lee, Y.-P. Hung, and J. Hsu, "Gesture-based interaction for a magic crystal ball," in *Proceedings of the 2007 ACM symposium on Virtual reality software and technology*, ser. VRST '07. New York, NY, USA: ACM, 2007, pp. 157–164.

[14] D. O. Gorodnichy and A. Yogeswaran, "Detection and tracking of pianist hands and fingers," in *Proceedings of 3rd Canadian Conf. Computer and Robot Vision*, 2006.

[15] R. Anchini, C. Liguori, V. Paciello, and A. Paolillo, "A comparison between stereo-vision techniques for the reconstruction of 3-d coordinates of objects," *Instrumentation and Measurement, IEEE Transactions on*, vol. 55, no. 5, pp. 1459 –1466, oct. 2006.

[16] C.-H. Su, H.-S. Chiu, and T.-M. Hsieh, "An efficient image retrieval based on hsv color space," in *Electrical and Control Engineering (ICECE), 2011 International Conference on*, sept. 2011, pp. 5746 –5749.

[17] "Lifecam HD5001," http://www.microsoft.com/hardware/en-us/p/lifecam-hd-5001/GNF-00001, Accessed January 22, 2013.

[18] "OpenCV," http://opencv.willowgarage.com/wiki, Accessed January 22, 2013.

[19] "Vitruvian Man," http://en.wikipedia.org/wiki/Vitruvian_Man, Accessed January 22, 2013.

[20] "Related video 1," http://www.youtube.com/watch?v=UHiQKcs-pII, Accessed January 22, 2013.

[21] "Related video 2," http://www.youtube.com/watch?v=XE1vU8F4Gec, Accessed January 22, 2013.

[22] "Related video 3," https://www.youtube.com/watch?v=i59v9U4bePk, Accessed January 22, 2013.

Longyu Zhang received the B.E degree in Telecommunications Engineering from Tianjin Polytechnic University, Tianjin, China, in 2010 and then the M.A.Sc degree in Electrical and Computer Engineering from University of Ottawa, Canada, in 2012. He is currently an engineer in the Multimedia Communications Research Laboratory, University of Ottawa.

His research interests include haptics and computer vision.

Jamal Saboune received his Engineering Diploma in Electrical Engineering from the Faculty of Engineering of the Lebanese University in 2002 and a Masters degree in Computer Sciences from the Universite de Technologie de Compiegne, France in 2003. He pursued his research works at the INRIA Lab in Nancy France and obtained his PhD in 2008. From 2009 till 2010 he held a post-doc position at the VIVA Lab, University of Ottawa and since 2011 he is a member of the Discover Lab at the University of Ottawa. His research interests include Computer Vision, HCI, Haptics, Artificial Intelligence and Tele-Medicine.

Abdulmotaleb El Saddik (F' 2009) is Distinguished University Professor and University Research Chair in the School of Electrical Engineering and Computer Science at the University of Ottawa. He held regular and visiting positions in Canada, Spain, Saudi Arabia, UAE, Germany and China. He is an internationally-recognized scholar who has made strong contributions to the knowledge and understanding of multimedia computing, communications and applications.

He has authored and co-authored four books and more than 400 publications. Chaired more than 40 conferences and workshop and has received research grants and contracts totaling more than $18 Mio. He has supervised more than 100 researchers. He received several international awards among others ACM Distinguished Scientist, Fellow of the Engineering Institute of Canada, and Fellow of the Canadian Academy of Engineers and Fellow of IEEE and IEEE Canada Computer Medal.