Vis Comput (2010) 26: 311–323
DOI 10.1007/s00371-010-0419-0

ORIGINAL ARTICLE

A color-based face tracking algorithm for enhancing interaction with mobile devices

Abdullah Bulbul · Zeynep Cipiloglu · Tolga Capin

Published online: 30 January 2010

Abstract A color-based face tracking algorithm is proposed to be used as a human-computer interaction tool on mobile devices. The solution provides a natural means of interaction enabling a motion parallax effect in applications. The algorithm considers the characteristics of mobile use-constrained computational resources and varying environmental conditions. The solution is based on color comparisons and works on images gathered from the front camera of a device. In addition to color comparisons, the coherency of the facial pixels is considered in the algorithm. Several applications are also demonstrated in this work, which use the face position to determine the viewpoint in a virtual scene, or for browsing large images. The accuracy of the system is tested under different environmental conditions such as lighting and background, and the performance of the system is measured in different types of mobile devices. According to these measurements the system allows for accurate (7% RMS error) face tracking in real time (20–100 fps).

Keywords Face tracking · Human computer interaction · Mobile devices · Motion parallax

1 Introduction

Recent advances in computer and mobile technology have made it possible to investigate new and efficient mobile interaction techniques. As a result, methods such as gesture recognition, motion and object tracking, perceptual user interfaces are gaining interest. As the input modalities are limited on a mobile device, the available resources should be used in an efficient and effective way. Keypad, joystick, stylus, camera, and sensors are some of the input sources that are commonly used for interaction.

In this paper, we focus on the usage of camera input for interaction with the device. We propose to use the built-in camera to track the head position to allow for a motion parallax effect in interaction. Thus, our approach does not require any extra special hardware. In this method, we track the head movements by comparing the face positions through the neighboring frames. Since the interaction technique will be used on a mobile device, and share the computational resources with the actual application, the solution should be real-time and should not consume many computational resources. Another challenging point due to mobility is the highly varying properties of camera input data, such as color, contrast, luminance, background, and ambient properties.

By using our method, different high-level gestures can be defined for different applications. Face tracking also enables enhancing the depth effect in the applications by supplying motion parallax. The motion parallax cue refers to the depth information provided by the optic flow of the visual field due to the sideways movement of the viewer [16], and is one of the most significant visual cues for conveying depth in a scene.

In order to demonstrate the usage of this face tracking system, we have implemented several sample applications. These applications demonstrate the motion parallax effect by using the face tracking system to control the camera in 3D applications, or enable the user to browse large images

A. Bulbul (✉) · Z. Cipiloglu · T. Capin
Department of Computer Engineering, Bilkent University, Ankara, Turkey
e-mail: bulbul@cs.bilkent.edu.tr

Z. Cipiloglu
e-mail: zeynep@cs.bilkent.edu.tr

T. Capin
e-mail: tcapin@cs.bilkent.edu.tr

by head movements that are mapped to scroll events in an intuitive way.

2 Related work

In the literature, face detection, face tracking and related computer vision techniques have been used for human-computer interaction. There are various proposed interaction approaches based on the mobile device's egomotion; i.e. tracking the self movement of the device. Among them, Barnard et al. propose a vision-based user interface for mobile devices, where the camera input is used for egomotion calculation which characterizes the motion of the user's hand [1]. In this method, Hidden Markov Models are used to model the motion feature sequences. Then, the results are filtered by a likelihood ratio and the entropy of the sequence. Capin et al. and Haro et al. suggest camera-based interaction solutions, in which the incoming video is used to estimate the phone motion and then the physical motion of the phone is mapped to scroll or view direction according to the application. In these methods, feature-based tracking algorithms are performed to analyze the motion of corner-like features between the consequent frames [5, 10]. Another feature-based approach for controlling user interfaces on mobile devices has been proposed by Hannuksela et al., where the motion analysis is performed using a sparse set of features, and a Kalman filter is applied to smooth motion trajectories [9].

Finger tracking by camera has also been addressed for user interaction on mobile devices. Hannuksela et al. combine Kalman filter and the Expectation Maximization (EM) algorithm for estimating the background and finger motions in order to deal with the varying background conditions [8].

Face tracking is an important vision-based tool for enhancing human-computer interaction. In these solutions, the motion of the face is translated into a set of commands to interact with the computer. In order to track the face position, the first step is to detect the face in an image. Face detection algorithms are generally categorized as feature-based and image-based methods [11]. In feature-based methods, facial features such as nose, eyes, and lips are identified by performing a geometrical analysis on their locations, proportions, and sizes [2]. Color, motion, and edges are the mostly used properties to extract the facial features. The second type of solutions, which contains image-based methods, is based on scanning the image through a window to find the face candidates. The methods in this category generally use template matching, support vector machines (SVM), or neural networks [13, 17]. Image-based methods are popular in tracking due to their robustness.

One example of feature-based approaches for face tracking is proposed by Bradski, where face tracking is used to control computer games and 3D navigation [2]. One of these applications is flying over a 3D city model controlled by face movements. This solution extends a color-based tracking method, called the mean shift algorithm, which operates on color probability distributions. Hunke et al. also propose an image-based neural networks algorithm that performs face tracking for human-computer interaction [13]. The Blink and Click project is another approach, where the traditional mouse is replaced by human facial actions [17]. For instance, the nose location is used as the mouse pointer and left/right eye blinks correspond to left/right mouse clicks. This algorithm is a combination of feature-based and image-based approaches.

Several other researchers have proposed vision-based solutions for interaction. We refer the reader to reference [14] for a further survey of these techniques.

The face tracking methods that have been listed are designed to operate on desktop or high-end mobile systems. In addition, these algorithms generally perform multiple passes over the image and require complex machine learning processes. Hence, they are computationally heavy to be applied on smart phone mobile devices, and there is a need for less expensive and lightweight solutions for use in combination with mobile applications. Additionally, these techniques generally have inferior results with the mobile use case, due to a changing background and lighting conditions when the user is moving. Furthermore, they disregard the characteristics of the mobile use case, which enable the face tracking process to bypass several common stages to simplify the tracking process. For instance, the front camera of a mobile device is generally directed towards its single user. Owing to these issues, an initial attempt is performed in one of our previous studies [4]. In this paper, we extend the previous algorithm to include face tracking in 3D. Moreover, the algorithm is refined by considering the features related to the face shape, and the performance is improved by restricting the region on which the algorithm is executed.

3 Approach

As a human-computer interaction solution, we propose a mobile head tracking system, which uses the input from the front camera to track the head position. In this system, we can make the assumptions that there is a user in front of the display and there is only one user most of the time; the most likely scenario for mobile devices. We target a lightweight face tracking algorithm that is suitable to use in mobile devices with limitations in terms of CPU power. Even for high-end devices, the speed of the algorithm is important, since the face tracking system shares the resources with the actual application, and causes an additional load when it is used

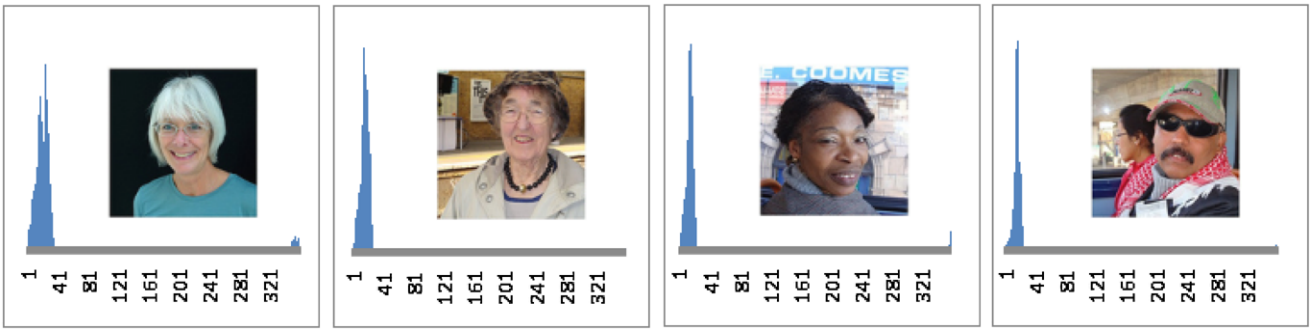

Fig. 1 Histograms of hue values across different people. The x-axis shows the hue values in 0–359 range. (Pictures are courtesy of Creative Commons (http://www.flickr.com/photos/joyoflife/3019568224/))

for interaction. Another important property of our solution is that it works under different lighting and background conditions.

Our camera-based face tracking system can be used as a general purpose interaction technique for various applications on mobile devices. For instance, it is suitable to be used for navigating in a 3D virtual environment, or adding motion parallax to rendering. Sample applications of the face tracking system are shown in Sect. 6.

3.1 Considerations and limitations

First, we list our assumptions for the tracking algorithm below.

- There is a single face on the image most of the time. The camera input generally shows the user in a head-and-shoulders view, and is at maximum distance of arm's length from the user.
- The face covers a large part of the image space when it is fully in view. According to our observations on a central horizontal line of the image, the facial region over the line is generally slightly above half of the complete line length.
- In a mobile environment, light is a highly varying factor, whereas the hue value of the face remains relatively stable throughout the frames. The hue values belonging to skin colors of different people fall in a particular range (Fig. 1).
- The color values will not fluctuate significantly, while going on a line on the face (Fig. 2).
- The background changes from frame to frame on a mobile environment, particularly when the user is on the move. Therefore, the mobile face tracking algorithm should be robust to changes in the background.

The limitations of mobile devices are also important for our algorithm design. Most importantly, smart phones have significantly lower computational power than desktop or mobile PCs. Our aim is keeping the face tracked in real time

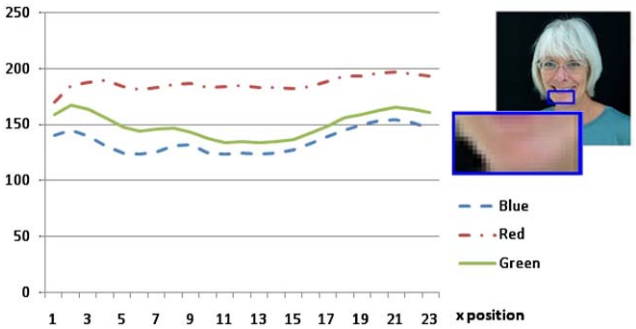

Fig. 2 RGB values through a line of a face. (Picture is courtesy of Creative Commons (http://www.flickr.com/photos/joyoflife/3019568224/))

while background properties change. On the other hand, existing vision techniques, particularly those based on features and edges, have inferior results in the mobile use case, due to changing background and lighting conditions when the user is moving.

Upon these design considerations, our algorithm is based on the following ideas:

- The algorithm is based on scanning only a limited region of the input image, rather than scanning the whole image. For example, it should be possible to use only a horizontal and a vertical slice of the image that includes the face.
- Slices to scan may be determined according to the previous position of the face, because of the temporal coherency of the face position through the frames.
- A scalable solution is preferred that allows for adjusting the preference over performance or accuracy according to application requirements.
- Feature-based and edge-based tracking techniques are not appropriate in our case, due to changing lighting and background, as well as the limited camera properties.
- At start, we assume that the face is close to the center of the image.

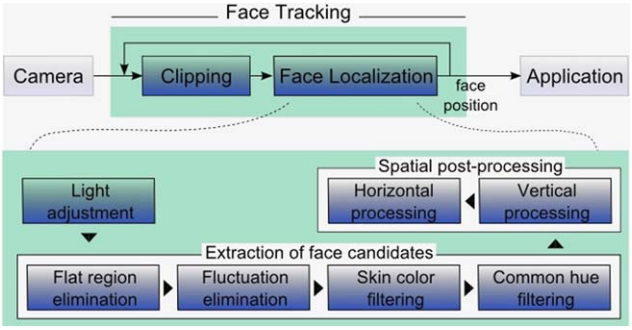

Fig. 3 Overview of the algorithm

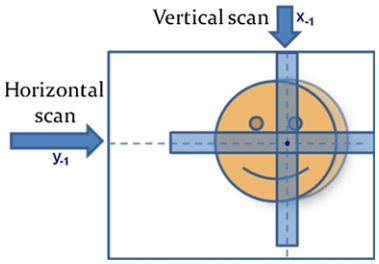

Fig. 4 Clipping phase of the algorithm. x_{-1} and y_{-1} represent the x- and y-coordinates of the previous face position

4 Algorithm

We have developed an algorithm that mainly uses color comparisons, rather than geometric properties of the facial features in order to avoid high complexity.

A color image can be represented in different possible color space models. The *RGB color space* is a common hardware-oriented color space; however, it is not an effective choice for tracking the human face due to two major reasons [7]: Firstly, in this color space, the brightness is not decoupled from the color information of the image. Secondly, the components should be normalized or processed in parallel to detect the human skin hue colors, as in Hunke and Waibel's work [13].

The *HSL color space* is more compatible with human color perception [15], and has been used for face extraction problems [2, 15]. In this space, the *hue* component represents the pure color; the *saturation* component represents the amount of white light mixed with the hue value; and the *lightness* component represents the brightness of the image. Thus, the HSL color space decouples *lightness* from color information. Furthermore, the *hue* component allows for discriminating color information for modeling skin color. Therefore, we have adopted the HSL color space, as the hue value is the stable parameter against the varying environmental conditions.

Our face tracking algorithm for user interaction is summarized in Fig. 3. In this approach, the image is first captured from the front camera of the device, then a suitable region of the image is selected (clipping phase) according to the previous face position, and the face is detected in the selected region of the image. According to the position of the head, an action specific to the application is performed, such as scrolling the picture to the right when the head moves to the left in a picture viewer application. We describe the details of our solution below.

4.1 Clipping

Since scanning through the whole image for every frame is costly on the mobile device, we have designed an algorithm in which all calculations are performed over a suitable small region of the image. Instead of tracking the regions corresponding to the facial regions as in previous approaches, we track a horizontal and a vertical region (Fig. 4). While determining the regions that will be scanned, we note:

- Which horizontal line or lines to use in calculations is determined by the y-position of the face in the previous frame, and vice versa.
- A scan line is not used as a whole; instead, a portion of it is used. This portion is determined according to the previous z-value which indicates how close the user is to the device. For example, as the user moves further from the device, his face covers a smaller area on the image and the scan line segment to consider becomes shorter, for acceleration. In our case, a scan line segment covers twice of the length of the previously detected face.
- The number of scan lines can be increased for more accurate results. This allows for a trade-off between accuracy and performance.

4.2 Face localization

After determining the horizontal and vertical lines to be scanned in the clipping phase, for each scan line, the following steps are applied.

4.2.1 Light adjustment

In a mobile environment the light is highly varying: the device can be used under sunlight, indoor environment, in a dark environment or in any different lighting conditions. Hence, the lightness values on the image may be in a narrow interval, such that in a dark environment all pixels on the image have low lightness values. A wider distribution of the lightness values is more preferable to work on. Thus, our localization algorithm starts with a light adjustment step.

First of all, in very low lightness levels, hue values cannot be correctly identified [2], and thus succeeding steps of the face localization algorithm cannot work correctly. Therefore, when the average lightness of the image falls under a

Fig. 5 *Left*: original image, *right*: light adjusted image

Fig. 6 *Left*: light adjusted image, *right*: flat regions are eliminated

specific value, all of the pixels having greater lightness values than the average are assumed to be on the face (1); as the only illumination source is the mobile device display which enlightens the face only, in most cases.

$$\forall p \in P, \quad (Avg < th) \wedge (p_L < Avg) \quad \Rightarrow \quad \text{eliminate} \qquad (1)$$

where P is the set of all pixels in the clipped region, p_L is the lightness value of pixel p, Avg is the average lightness value of the pixels in P, and th is the threshold value for minimum average lightness value for correct identification of hue values. In our case, th is chosen as 10 empirically.

When the average lightness value allows for extracting the hue value correctly, the light adjustment step is performed. The lightness values are gathered using the HSL color space, and after refinement of the L values, new HSL values are obtained based on the old ones (2).

$$p_{L'} = \begin{cases} p_L \times Aim/Avg, & p_L < Avg \\ 255 - (255 - p_L) \times \frac{255-Aim}{255-Avg}, & p_L \geq Avg \end{cases} \qquad (2)$$
$$\forall p \in P,$$

where P is the set of all pixels in the clipped region, $p_{L'}$ is the new light value of the pixel p, p_L is the old value, Avg is the average light value of all pixels in P, Aim is the target average lightness value of the region and the lightness value is scaled between 0–255. The purpose of this adjustment is setting the average light of the image to a central value and widening the light distribution (Fig. 5).

4.2.2 Extraction of face candidates

The steps in this part of the algorithm extract the pixels that potentially correspond to the face, which is achieved by eliminating the pixels that cannot be considered as a face based on their color properties. This part consists of four steps, which are explained below.

Flat region elimination This step in our algorithm eliminates the regions whose light values do not change for a long sequence of pixels, due to the curvature property of the human face. Pixels corresponding to a flat shape, for example a wall, have light values that are nearly the same since each part of a flat surface gets the light in a similar angle. On the other hand, the human face is not flat and has a curvature; thus, the light values on corresponding frames are expected to differ slightly on a face. This part of the algorithm sums up the differences in a sequence of pixels and eliminates them if the change in light values is below a threshold (3).

$$\forall p \in P, \quad \sum_{n \in N(p)} |n'_L| < th \quad \Rightarrow \quad \text{eliminate} \qquad (3)$$

where P is the set of all pixels in the clipped region, $N(i)$ is the set of neighboring pixels of pixel p, n'_L is the derivative of the lightness value of pixel n in HSL space, and th is a threshold value. We have empirically chosen a threshold of 25 for 30 consequent pixels (in 0–255 light value scale). Figure 6 illustrates this step.

Fluctuation elimination The next step of the algorithm is the elimination of the fluctuating regions. On the image, there may be a few textured parts that have a pattern of repeating colors or a non-regular region that have very different colors. These parts cannot be on a face, since the color values on a face generally change monotonically; thus, there is not much fluctuation of the color values on a face. Using R, G and B together is more suitable than using H, S and L in this step, since R, G and B are of similar types in terms of range and semantics. Thus, we use the RGB space in this step. Determining the fluctuating parts is done by taking the second order derivatives of the RGB values. Second order derivatives for R, G and B components are separately calculated and the magnitudes of the derivatives are summed up. If the summation of the fluctuation is greater than a threshold in a line of consequent pixels, this part is eliminated (4).

$$\forall p \in P, \quad \sum_{n \in N(p)} |n''_r| + |n''_g| + |n''_b| > th \quad \Rightarrow \quad \text{eliminate} \qquad (4)$$

where P is the set of all pixels; $N(i)$ is the set of neighboring pixels of pixel p; n_r, n_g, n_b are the RGB components of pixel n; and th is the threshold value. In our case, the threshold is 750 for 30 consequent pixels (in a 0–255 RGB scale). This part of the algorithm uses a similar approach to edge detection techniques, and can be seen as elimination of the parts that have many edges (Fig. 7).

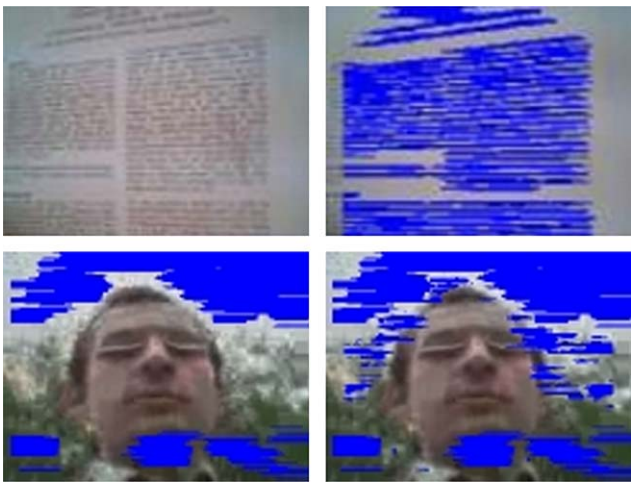

Fig. 7 *Top*: effect of fluctuation elimination (*left*: original, *right*: fluctuations are eliminated), *bottom*: fluctuation elimination step in our system. (*Left*: before fluctuation elimination, *right*: after fluctuation elimination)

Skin color filtering There are a number of prior attempts to detect the skin color, and most of them suggest very restrictive methods [18]. These restrictions are not appropriate for mobile environment, since they cause most of the face to be eliminated in the varying environmental conditions. Therefore, we use a less restrictive method to detect the skin color.

The observations of Brand and Mason [3] suggest that the blue component in a skin color varies in a wide range, hence blue component is not effective on the overall color as much as the red component. According to the analytical assessments of the same work [3], the red-green ratio of the skin color changes between 1:1 to 3:1. Based on these assessments, we eliminate the regions with color that cannot be a skin color (5) (Fig. 8).

$$\forall p \in P, \quad \neg\left(1 < \frac{p_r}{p_g} < 3\right) \quad \Rightarrow \quad \text{eliminate} \tag{5}$$

where P is the set of all pixels, p_r and p_g are red and green components of pixel p.

Common hue filtering One assumption is that the face generally covers a large region in the display in a mobile device. The proportion of the face to the whole is generally above $1/2$ and at least $1/3$ in each dimension. Based on this assumption, the mostly used hue value will likely belong to the face. For this purpose, we divide the hue space into equal sized clusters, and for each cluster, we find the total number of pixels that fall into that range. The cluster with the maximum number gives the mostly used hue and we assume that this is the dominant hue value on the face. Therefore, we can eliminate other regions in the image. A cluster size of 40 generally gives the best results (Fig. 9).

Fig. 8 *Left*: before skin color filtering, *right*: after skin color filtering

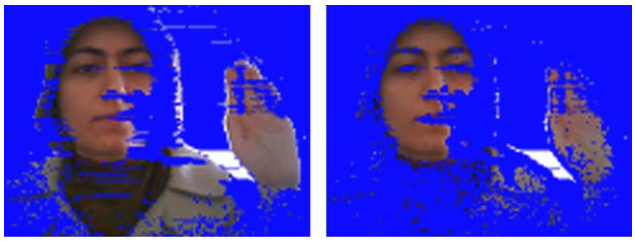

Fig. 9 *Left*: before common hue filtering, *right*: after common hue filtering

4.2.3 Spatial postprocessing

After eliminating the pixels that cannot belong to the face, the remaining pixels are analyzed according to their spatial properties, to localize the actual face among them. The coherency of the pixels belonging to a face can be used for this purpose, following Hunke and Waibel's approach [13]. In this phase, our aim is to eliminate the non-coherent pixels, due to the coherency of the face shape. This part consists of the following two steps.

Vertical processing While scanning the image horizontally; if the majority of the pixels on different scan lines and on the same x-position are determined as face candidates, they have a high possibility of belonging to the face. Hence, this step of the algorithm eliminates the pixels, if the majority of the rest of the pixels on the same vertical line are eliminated previously (6). Similarly, while scanning the image vertically, the number of the candidate pixels with the same y-position is considered. This step of the algorithm is not effective when only a single scan line is used.

This step of the algorithm is implemented as follows. On a horizontal scan, for each x-position, let p_x be the probability of the face existing in this x-position. p_x is calculated as follows:

$$p_x = \frac{\text{number of lines in which } x \text{ belongs to face}}{\text{number of scan lines}} \tag{6}$$

If p_x is smaller than a threshold, the xth pixels of each scan line are eliminated. In our case, the threshold is $\max(p_x)/2$, where $\max(p_x)$ is the probability of the pixel with the highest probability in this scan (Fig. 10).

Horizontal processing In this step, the intensity of the face candidates in a horizontal line is considered. The main idea behind this step is that when we form several isolated groups of candidate pixels on a scan line according to their closeness, the largest group belongs to the actual face, and others should be eliminated.

This step is defined as follows. In a scan line, a counter is kept, which is incremented when a candidate pixel is encountered and decremented when an eliminated pixel is encountered, without allowing it to be negative. If the counter changes from 0 to 1, a new group is started and when it decreases down to 0 the group is completed. Among these groups, the group with the highest counter value is selected to be the actual face and the candidate pixels in the other groups are eliminated (Fig. 11).

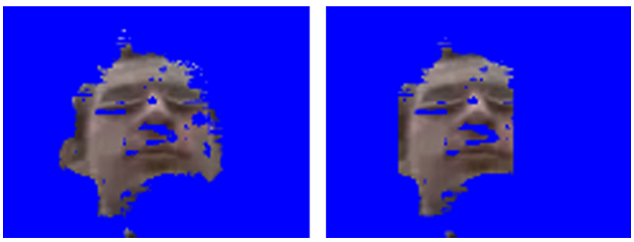

Fig. 10 *Left*: before vertical processing step, *right*: after vertical processing step

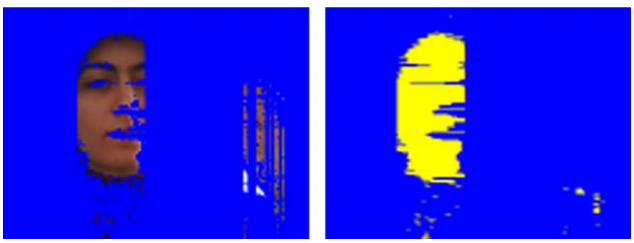

Fig. 11 Effect of horizontal processing step; *left*: before, *right*: after. *Yellow pixels* indicate the facial region. As seen in the figure, the user's hand (noise on the *right side* of the *left* image) is eliminated in this step

4.2.4 Determining the face position

After all of the previous steps, whose effects are shown in Fig. 12, the 3D face position is determined according to the non-eliminated pixels. The x- and y-positions are calculated as the average of the x- and y-coordinates of the non-eliminated pixels and z-value is extracted from the proportion of the number of non-eliminated pixels to the number of all pixels (7):

$$x, y = \frac{\sum_{f \in F} f_{x,y}}{|F|}, \qquad z = \frac{|A|}{|F|} \tag{7}$$

where x, y, and z are the normalized values of the detected face coordinates; A is the set of all pixels and F is the set of all facial pixels.

5 Results

5.1 Accuracy

We have tested the accuracy of our face tracking system using 8 video sequences, which have 120×90 resolutions, representing different mobile use cases. These cases include environmental variability and different users in various states such as walking, standing and traveling in a bus.

The plots in Fig. 13 compare the face positions that are found by the face tracker, with the real positions in x- and y-coordinates through different motion sequences. The real positions are obtained by manually marking the center of the eyebrows offline for each frame.

In order to measure the accuracy of our face tracking solution, we calculate the root mean square error (RMS) of the tracker outputs relative to the real face positions for x- and y-dimensions separately. The RMS is calculated as follows:

$$\mathrm{RMS}(T) = \sqrt{\frac{\sum_{i=1}^{|T|} (T_i - F_i)^2}{|T|}} \tag{8}$$

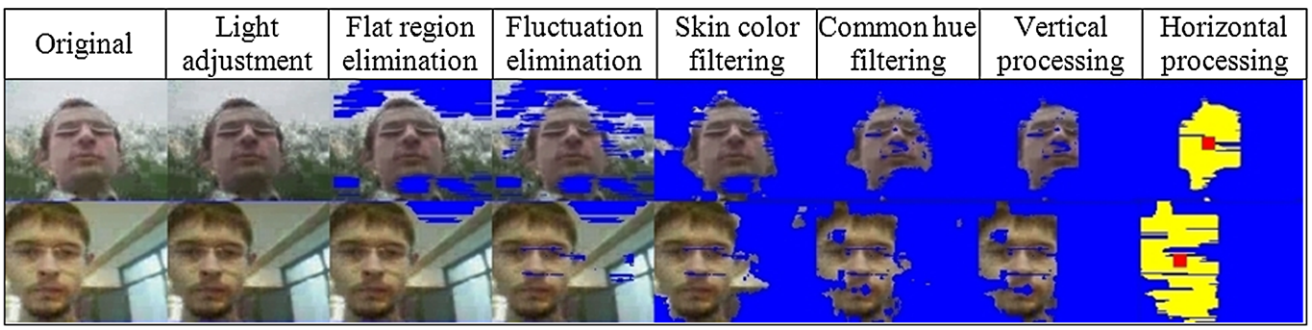

Fig. 12 Step by step execution of the algorithm on two different samples. (For a better demonstration of the steps, the whole image is scanned without clipping)

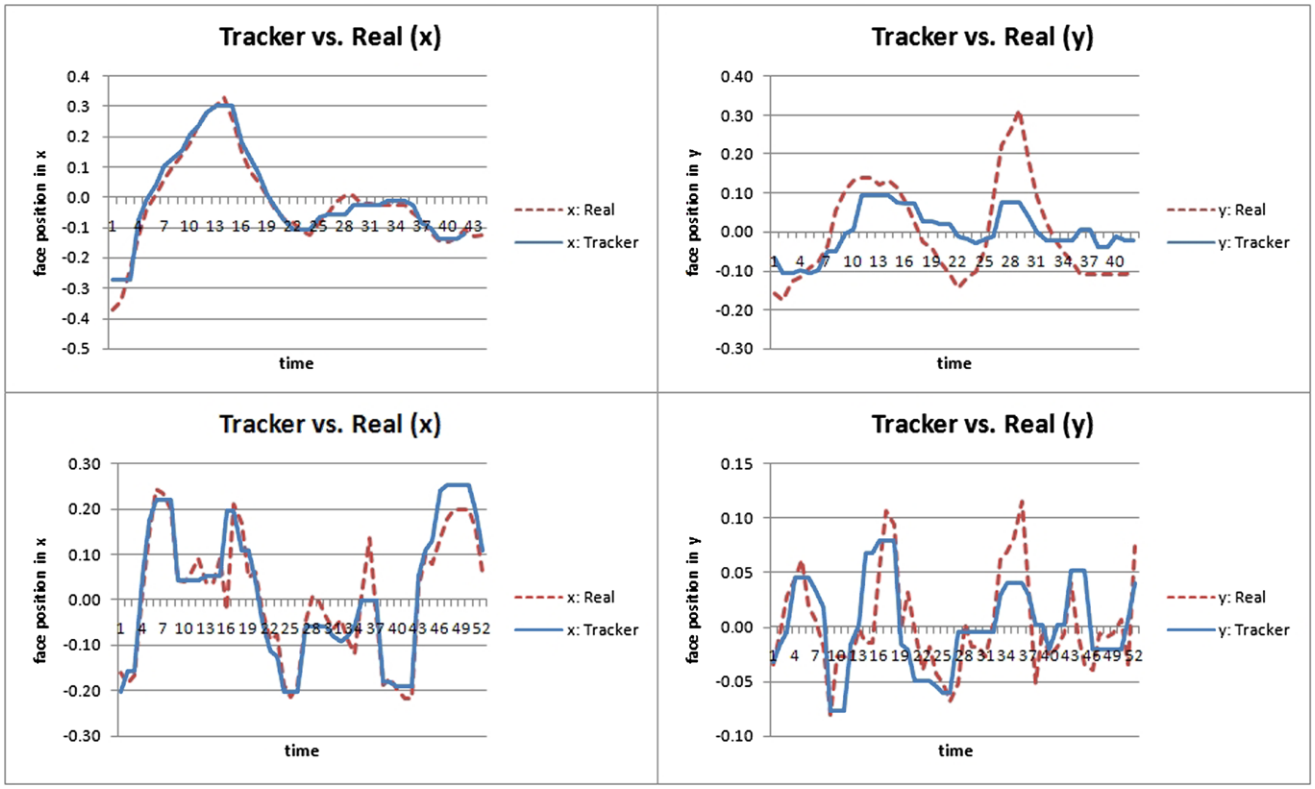

Fig. 13 Tracking results compared to real positions, in x- and y-dimensions, for sample motion sequences. (The screen is mapped into $[-0.5, 0.5]$ interval from *left* to *right*)

where T is the set of tracker outputs and F is the set of real face positions. Figure 14 plots the average RMS errors of all test sequences according to the number of scan lines in each direction.

According to the measurements, the system has an RMS error rate of about 7% on average. Even though the RMS error is a significant indicator of the system accuracy, in a tracking system, tracking the relative motion of the face is sufficient in the described mobile cases, as opposed to detection of absolute face location. For example, in the top-right plot of Fig. 13; although the response of the system to the user is in the correct direction, the difference between the real and the tracker generated face positions is high, which in turn propagates the measured error.

The accuracy of the system is acceptable even when a small number of lines are scanned (Fig. 14). When more lines are scanned, up to a point, the accuracy of the system increases in general; then, the accuracy of the system becomes generally stable.

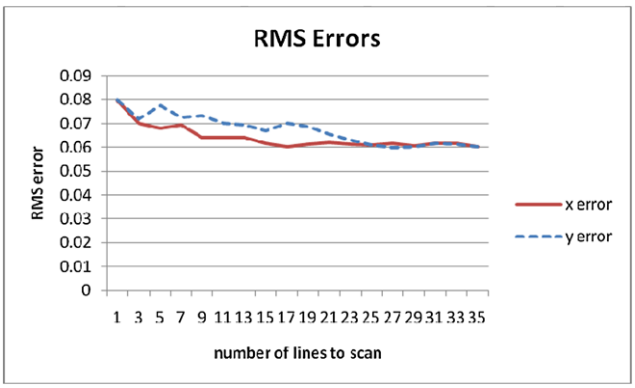

Fig. 14 Average RMS errors for x- and y-dimensions

5.2 Performance

We have tested the performance of our face tracking algorithm on different mobile platforms, including Nokia N95 Mobile Phone, Sony Vaio UMPC, and Acer 5920G notebook. In addition to the device capabilities, there are two major factors that affect the performance of the algorithm: the number of lines that are scanned, and the image capture resolution. In order to demonstrate the effects of these variables, we have measured the performance of our system on different devices, using various image resolutions and different number of scanned lines.

The results of performance measurements are shown in Fig. 15, in which the execution time of our algorithm is plotted against the different number of scanned lines. Figure 15 shows that an increase in the number of scanned lines results

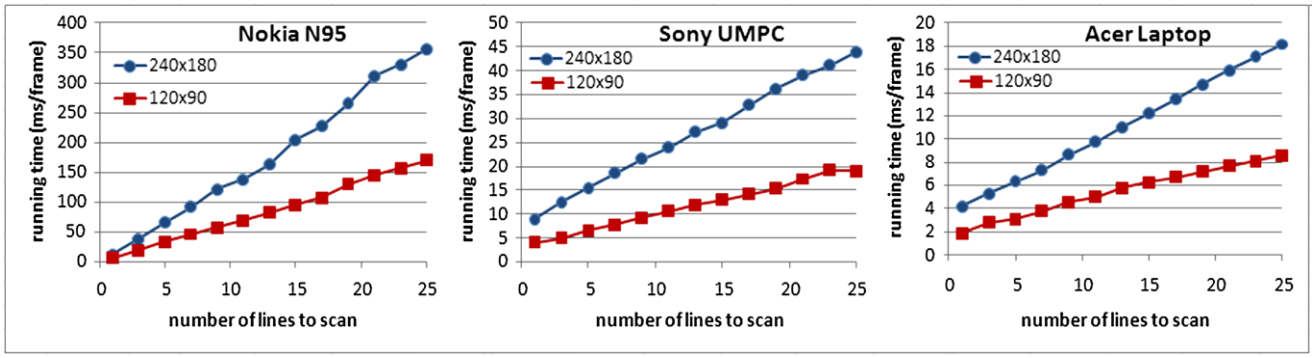

Fig. 15 Performance results. Two different image resolutions are used: 240 × 180 and 120 × 90 for each device. Running time is the time required to execute the algorithm for one frame in milliseconds

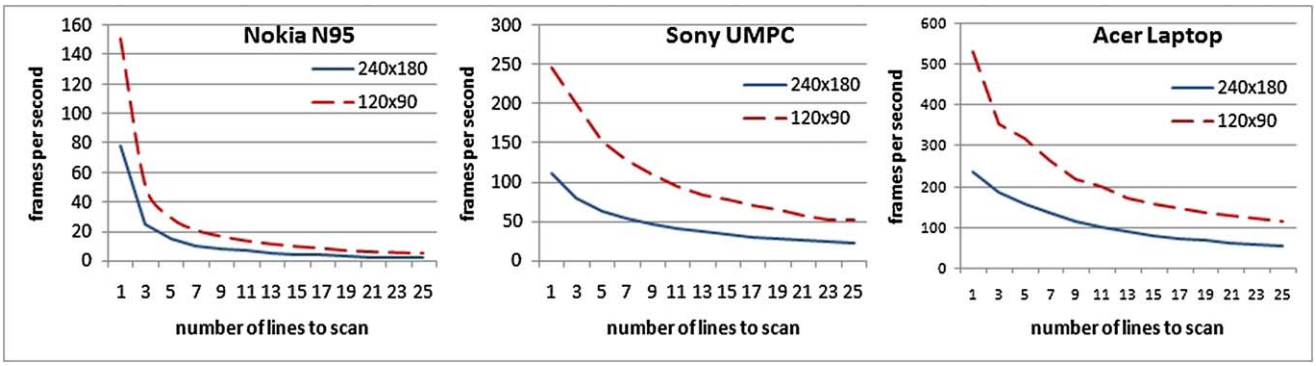

Fig. 16 Performance results in terms of fps. (Two different image resolutions are used: 240 × 180 and 120 × 90 for each device)

in a linear increase in the running time of the algorithm, as expected. Moreover, using quarter sized images, which actually reduces the number of scanned pixels by half since the number of lines to scan remains constant, decreases the running time of the algorithm to half of the original. This result reflects that our algorithm runs in linear time with respect to the number of scanned pixels.

Note that the accuracy results in the previous section suggest that using an image resolution of 120 × 90 is sufficient for approximate results, and it also suffices to scan a small number of lines. Thus, although the accuracy-performance trade-off changes according to the application requirements, it is more preferable to use low-resolution images (120 × 90) and scan small number of lines, which maintain the accuracy in acceptable levels. In Fig. 16, performance results in terms of frames per second are plotted for each device. On the average, the algorithm runs with 22, 128, and 267 fps for Nokia N95, Sony UMPC, and Acer notebook respectively; when 7 lines are scanned using low-resolution images. It is also possible to increase the performance notably by sacrificing a little accuracy.

One important point is that our algorithm needs to process only a part of the image, but in the platforms we have used, the whole image is still read from the camera input. Hence, a driver-level implementation, which allows for getting only the required part of the image from the camera, would improve the performance of our algorithm significantly.

6 Applications

Integration of face tracking into applications significantly enhances the means of interaction in mobile devices, which lack common interaction modalities such as mouse-based interaction. First, face tracking should not be considered as a substitute of motion sensor-based interaction. An accelerometer is generally more accurate and more efficient than face tracking; however, the real contribution of the face tracker is different. The tracker provides an intuitive interaction way for specific applications by enabling them to respond the natural expectations of a user according to his eye position. A perceptual phenomenon, called motion parallax, is enabled by the use of face tracking in applications. Motion parallax refers to the optic flow of the visual field due to the sideways movement of the viewer [16], and is one of the most significant visual cues for enhancing depth perception in a scene. Face tracking provides motion parallax when the viewpoint is determined according to the users face position in an application.

Motion parallax has a strong effect on simplifying the following tasks by providing additional depth information [19]:

- Tracing data paths in 3D graphs.
- Judging the morphology of surfaces.
- Interpreting patterns of points in 3D space.
- Identifying relative position information.
- Reaching for objects.

Therefore, face tracking will be especially effective when the applications include any of these tasks. Another benefit of using face tracking is its contribution to the users' sense of presence. Accordingly, it would be useful to integrate a face tracking system into a number of applications such as: 3D scenes, games, CAD tools, image viewing tools, web browsers, etc. An additional usage of face tracking would be in Augmented Reality (AR) applications. In mobile device based AR systems, the user's position should be provided to the applications to better align the viewpoint. For this purpose, in the current AR applications hardware motion sensors and Bluetooth beacons are used [6]; which can be replaced by the face tracking system, eliminating the necessity of any extra hardware.

We present several applications that we integrated our face tracker into, in order to demonstrate example use cases of our interaction solution.

Camera application As a sample application that uses our face tracking algorithm, a 3D application was implemented (Fig. 17). In this application, the 3D scene is rendered from different viewpoints using an OpenGL perspective camera. The position of the camera is determined by the input from our face tracking system. In other words, the position of the user is used as analogous to the position of the camera and the movement of the camera is controlled by the movement of the head. In this way, the user can view different portions of the scene by only moving his head or his device, and this enhances the depth perception by providing the motion parallax depth cue.

Game viewpoint control helper application In this application, although the face tracking system is not used for the navigation and actions of the game character, it is used to control the head movements of the character in a first person shooter game (Fig. 18). For instance, a game character that

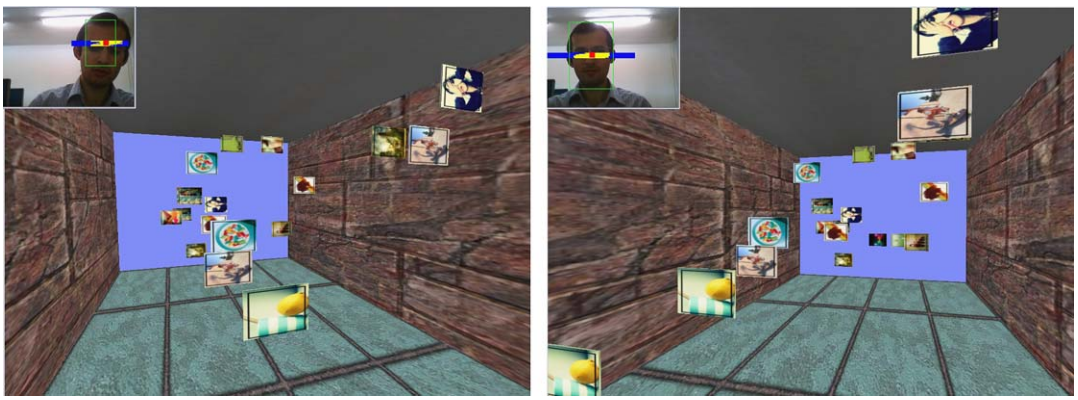

Fig. 17 Snapshots from the camera application. (*Left*: the user looks at the virtual scene from left. *Right*: the user looks at the scene from right)

Fig. 18 Snapshots from the game viewpoint helper application. The face tracking system is integrated into a sample game using Xith3D toolkit [12]. (*Left*: the user looks at the scene from left. *Right*: the user looks at the scene from right to see around the corner)

is hiding behind a wall should be able to see around corners in the scene by moving his head without revealing his entire body.

Image browser application Displaying a large picture in a small display area has always been a problem of mobile devices. The lack of mouse makes the usage of scroll bars difficult. For this purpose, an intuitive interaction solution is required. We use our face tracking system for handling the inputs for scrolling events. Therefore, we have implemented a system to view large pictures such as panoramic views in a small display (Fig. 19). The usage of this system is quite intuitive because it can be thought as looking through a window as a peephole metaphor [20]. Thus, when the user moves his head to the right, he can see the area on the left from the window, and vice versa. This usage is much easier in mobile devices when compared to desktop systems, since there is also an opportunity of moving the device as well as the viewpoint.

7 Discussion and conclusion

In this paper, we have proposed a new face tracking based interaction method for mobile devices. The proposed face tracking method is based on color comparisons and designed to be appropriate for the limitations of the mobile devices and the highly varying mobile environment.

The primary purpose of face tracking is to provide a natural user interaction modality by enabling motion parallax in specific applications. The usage of the face tracking is discussed in Sect. 6 and demonstrated with several example applications.

Generally the system works as expected in different environments and under different lightings, and even in a relatively dark environment, the user's face can be differentiated by the system to some extent. Figure 20 shows sample execution results under various light and background conditions. We have measured the accuracy of our solution by calculating the RMS errors of the tracker determined face

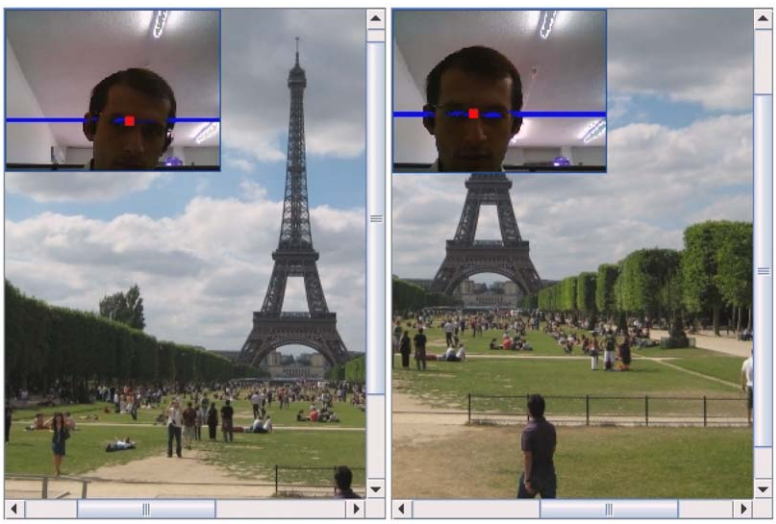

Fig. 19 Snapshots from the image browser application. (The user looks at the scene from two different viewpoints to browse different parts of a large image)

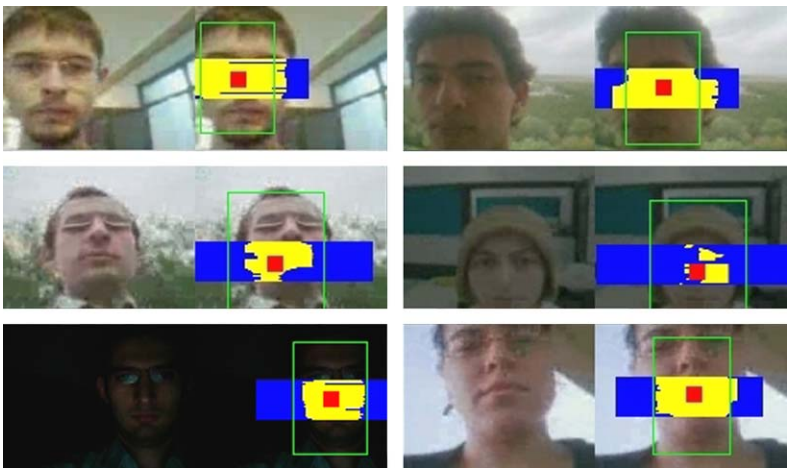

Fig. 20 Results under different background conditions. The *first* and *third columns* are the original images. The *second* and *forth columns* are the output of the face tracker. The algorithm runs on the clipped region which is depicted by a *blue rectangle*. (*Blue regions* are the eliminated parts, *yellow regions* are the facial pixels, the *red point* is the calculated face position, and the *green rectangle* reflects the depth information)

positions against the actual face positions. The accuracy of the system under different conditions is close to 93% on the average, as shown in Fig. 14. However, in order to evaluate the success of the face tracker, it is not necessary to track the absolute face movement; instead, tracking an approximate and relative position is sufficient for interaction. Hence, these accuracy results constitute an upper bound for the error rate of our face tracking system.

There are limitations of our algorithm: according to our observations, the system cannot be used appropriately in some conditions. Firstly, some flat wooden parts such as many of the furniture cannot be differentiated from the face easily, since they have similar hue values and they have texture that is not flat or fluctuating enough to be eliminated. Also, the system cannot work correctly when there is a white background that is illuminated by yellow or pink light which creates color properties similar to a face. In these cases, a color-based tracking algorithm does not suffice. A possible solution for these cases can be extending the algorithm to consider the geometrical features such as the proportions of nose, eyes and lips etc., in addition to the color-based calculations.

One advantage of the system is its adjustable usage, which provides flexibility needed for mobile devices with different CPU power and camera resolutions, as explained in Sect. 5.2. Since all calculations are done line by line in a horizontal or vertical fashion, it is possible to set the number of scan lines to consider. The proposed method is scalable: all of the image can be used in the calculations, as well as only a single vertical and a single horizontal line of the image. Since increasing the number of lines to scan does not increase the success of the system after a specific number (for example, 11 lines), scanning a large portion of the image is useless.

In conclusion, face tracking can be used as an intuitive and effective way of interaction. The proposed color-based algorithm targets overcoming the mobile environment challenges, such as background and lighting variety. Furthermore, the adjustable behavior of the proposed face tracking system can be a significant facility for mobile devices which have different computational capacities. In the future, our further research will improve the face localization part of the algorithm to obtain more accurate results. Moreover, a driver-level implementation of our system remains as a future work to improve the performance of the system.

Acknowledgement This work is supported by the European Commission FP7-213349 All 3D Imaging Phone project and TUBITAK.

References

1. Barnard, M., Hannuksela, J., Sangi, P., Heikkilä, J.: A vision based motion interface for mobile phones. In: Proc. of 5th International Conference on Computer Vision Systems (ICVS), Bielefeld, Germany (2007)
2. Bradski, G.R.: Computer vision face tracking for use in a perceptual user interface. Intel Technol. J. 95–103 (1998)
3. Brand, J., Mason, J.: A comparative assessment of three approaches to pixel-level human skin detection. In: Proc. of IEEE International Conf. Pattern Recognition, vol. 1, pp. 1056–1059 (2000)
4. Bulbul, A., Cipiloglu, Z., Capin, T.: A face tracking algorithm for user interaction in mobile devices. In: Proc. of Cyberworlds, International Conference, pp. 385–390 (2009)
5. Capin, T., Haro, A., Setlur, V., Wilkinson, S.: Camera-Based Virtual Environment Interaction on Mobile Devices, Lecture Notes in Computer Science, vol. 4263, pp. 765–773. Springer, Berlin (2006). ISBN: 9783540472421
6. Capin, T., Pulli, K., Akenine-Möller, T.: The state of the art in mobile graphics research. IEEE Comput. Graph. Appl. **28**(4), 74–84 (2008)
7. Chai, D., Ngan, K.N.: Face segmentation using skin color map in videophone applications. IEEE Trans. Circuits Syst. Video Technol. **9**(4), 551–564 (1999)
8. Hannuksela, J., Huttunen, S., Sangi, P., Heikkilä, J.: Motion-based finger tracking for user interaction with mobile phones. In: Proc. of 4th European Conference on Visual Media Production (CVMP). London, UK (2007)
9. Hannuksela, J., Sangi, P., Heikkilä, J.: A Vision-based approach for controlling user interfaces of mobile devices. In: Proc. of IEEE Conference on Computer Vision and Pattern Recognition, Workshop on Vision for Human-Computer Interaction (V4HCI), vol. 6, p. 71, San Diego, CA (2005)
10. Haro, A., Mori, K., Capin, T., Wilkinson, S.: Mobile camera-based user interaction. In: Proc. of ICCV-HCI 2005, pp. 79–89 (2005)
11. Hjelmas, B.K.L.E.: Face detection: a survey. Comput. Vis. Image Underst. **3**(3), 236–274 (2001)
12. Home of the Xith3D Project. Xith3D.org. http://xith.org/. Accessed 27 October 2009
13. Hunke, M., Waibel, A.: Face locating and tracking for human-computer interaction. In: Proc. of the 28th Asilomar Conf. on Signals, Systems and Computers, vol. 2, pp. 1277–1281 (1994)
14. Jaimes, A., Sebe, N.: Multimodal human computer interaction: a survey. In: Proc. of 11th IEEE International Workshop Human Computer Interaction (HCI) (2005)
15. Kakumanu, P., Makrogiannis, S., Bourbakis, N.: A survey of skin colormodeling and detectionmethods. Pattern Recognit. **40**, 1106–1122 (2007)
16. Shirley, P.: Fundamentals of Computer Graphics. AK Peters, Natick (2002)
17. Siriluck, W., Kamolphiwong, S., Kamolphiwong, T., Sae-Whong, S.: Blink and click. In: Proc. of the 1st International Convention on Rehabilitation Engineering & Assistive Technology: in Conjunction with 1st Tan Tock Seng Hospital Neurorehabilitation Meeting, pp. 43–46 (2007)
18. Vezhnevets, V., Sazonov, V., Andreeva, A.: A survey on pixel-based skin color detection techniques. In: Proc. of GRAPHICON03, pp. 85–92 (2003)
19. Ware, C.: Space Perception and the Display of Data in Space. Information Visualization: Perception for Design. Morgan Kauffman, San Mateo (2004)
20. Yee, K.P.: Peephole displays: Pen interaction on spatially aware hand-held computers. In: Proc. of CHI 2003, pp. 1–8 (2003)

Abdullah Bulbul is a Ph.D. candidate in the Computer Engineering Department of Bilkent University. He received his B.S. degree in the same department in Bilkent University in 2007. His main research area is Computer Graphics and in particular, the use of Perceptual principles in Computer Graphics.

Zeynep Cipiloglu is an M.S. student in the Department of Computer Engineering, Bilkent University. Her research interests are Computer Graphics, Perception in Computer Graphics, Mobile and Ubiquitous Graphics, and 3D User Interfaces. She is currently working on 3DPhone project by EC 7th Framework Programme.

Tolga Capin is an assistant professor at the Department of Computer Engineering, Bilkent University. Before joining Bilkent, he has worked at the Nokia Research Center as a Principal Scientist, where he led various graphics research activities. He has contributed to various mobile graphics standards, including Mobile SVG, OpenVG, JCP, and 3GPP. Tolga has received his PhD at EPFL (Ecole Polytechnique Federale de Lausanne), Switzerland in 1998. He has published more than 20 journal papers and book chapters, 30 conference papers, and a book. He has 2 patents and 10 pending patent applications. His current research interests include mobile graphics platforms, human-computer interaction, and computer animation.