A Head-Controlled User Interface

Vincent Graveleau, Nikolay Mirenkov, Gennadiy Nikishkov
Computer Software Department, The University of Aizu
e·mail: {m5082102, nikmir, niki}@u-aizu.ac.jp

Abstract

In this paper, we present an implementation of a head-controlled real-time human-computer interface that does not require using hands. This kind of interface may be valuable for disabled people or in cases where hands are busy with other tasks.

Usual video-camera is employed for tracking head movements. Position of the eyes is determined with the use of image analysis. Head movements are mapped to cursor displacements. Nodding and head-shaking model left and right mouse clicks. Experiments show that the system behaves satisfactory even for usual web-camera frame rates (15fps), and small images (320×240). It consumes small percentage of CPU resources allowing other processes to run smoothly.

Keywords: Face tracking, eye detection, image processing, connected component analysis, user interface, cursor mapping, head-control.

1 Introduction

Typically, the user interacts with a computer using a mouse and a keyboard. However, there are cases when the mouse-keyboard interface is impossible or inconvenient. People with certain disabilities cannot communicate with the computer using the keyboard or mouse. Head movement can help a disabled person to interact with the computer. Due to common availability of video-cameras and sufficient power of ordinary computers the head-controlled user interface can be implemented without employment of any special hardware. The head-controlled user interface may be useful in some other situations. For example, people driving cars can find convenient to control some devices (not critical for safety) by head movements.

To follow head movements, different features of a face can be tracked, like eyes, nostrils [1], or nose bridge [2]. Tracking of nostrils is more sensible to light variation and camera placement than eye-tracking. The nose detection method described in [2] is based on eye detection with subsequent search on the nose. Eye-tracking usually appears more stable than other techniques.

Several approaches have been investigated for detecting and tracking the eyes, such as infrared techniques [3], binocular eye-tracking [4] (purely optical), or EOG (electro-oculographic potential) analysis. Infrared techniques require infrared illumination and an infrared-sensitive video camera [3] to track eye movements. Binocular eye-tracking [5] is based on attaching electrodes next to the eyes and on using them to sense the EOG potential, which varies according to the eye gaze. Those systems require nonstandard expensive equipment, and are not suitable for our purpose. Thus we have selected to track the eyes using a simple but efficient connected component analysis of images registered by video-camera.

The eyes are a common feature to every human being, and relatively easy to track. For tracking head movements, the "Between-The-Eyes" (BTE) point is a good candidate. Experiments show that the BTE position is stable when the user is calmly facing the screen. This relaxed position will be used as the center of the cursor mapping.

Kawato et al. [6, 7] described a system to track the BTE point. Their approach requires extensive computations, but leads to an accurate BTE position. The present system objective is to make this BTE localization as fast as possible, sacrificing a bit of accuracy but increasing the efficiency of the frame processing. A low use of processor resources is needed for other process tasks, because our system should behave as a human-computer interface with low consumption of CPU power. It has to be noticed that later experiments have showed that the loss of precision is not significant for the head-controlled interface.

In addition to the fast BTE tracking system, a cursor mapping interface has been developed. Unlike previous works on the subject, this system makes an efficient use of the tracking data and maps smoothly the head movements to the cursor displacements, thus allowing the user to control the computer. Our system provides a powerful tool for situations where the hands are not available (such as car navigation displays, or in the case of disabled people unable to move the hands).

The structure of this paper is as follows. Next section details the face detection algorithm. The eye detection process, introducing the bisection thresholding method is presented in Section 3. The face movement interpretation, and the cursor mapping interface are described in Section 4. Finally, some experimental results are discussed.

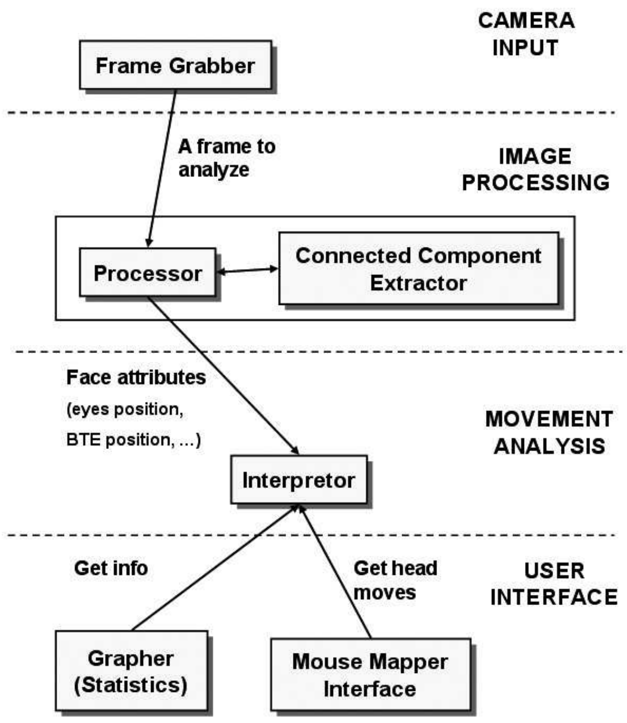

Figure 1: Overview of the head-controlled interface.

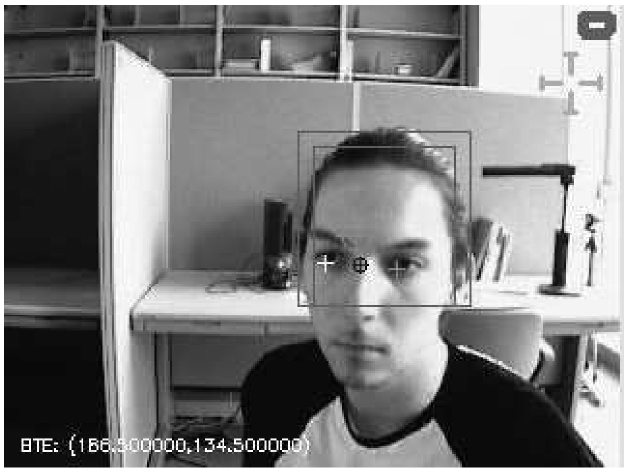

Figure 2: Face detection.

2 Overview of the System

The present head-controlled user interface works as described in Figure 1. Once the general interface is initialized, the main thread is started, and the processing loop begins. A frame grabber (web-camera, car camera, etc) grabs a frame, and passes it to the image processing unit of the system. The system starts in detection mode, and attempts to localize the face. If the localization is successful, the tracking mode begins. If not, the frame is discarded. The searching area for the eyes depends on the mode. In tracking mode, the area near the previous location is used, resulting in a great increase of performance. In detection mode, the face candidate area is used. When the eyes are detected the current frame attributes are stored. The interpreting unit employs attributes of several last frames to detect head movements. Head movement data is used in the cursor mapping interface, so that the cursor may be moved, and clicks may be performed.

3 Face Detection

The first step in the image processing is to extract a face candidate. This operation is done in detection mode only, by detecting the moving areas of a frame. Once a face candidate is found, as shown in Figure 2, the detection of the eyes may be done.

Adjacent frame subtraction, a known technique for extracting moving objects [8], is used here to find a face candidate. The pixel-wise subtrac-

a) b) c)
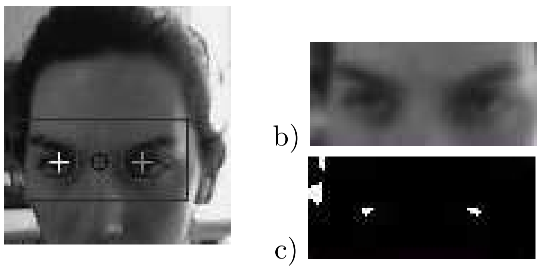
Figure 3: a) The processed image with the tracking area (rectangle); b) The smoothed grayscale area to search for eyes; c) The same area thresholded with the best value.

tion employs luminance data. To remove noisy blobs, the following steps are performed. First, we make a difference image: each pixel has an absolute value of the difference in luminance between current and previous frames. Then for each horizontal image line, we find the left-most pixel and the right-most pixel where the pixel values exceed some predefined threshold. The segment between the left-most and right-most pixels is assumed to be a moving segment. Scanning from the top line to the bottom, the first line with a moving segment having a width that is larger than a predefined threshold (expected smallest face width) is considered the top edge of a face candidate. Then we continue scanning down the lines while updating size and position of the rectangular area that includes all scanned moving segments until the area's height exceeds the expected maximum useful height of a face or until reaching the bottom end of the image.

4 Eyes Detection and Tracking

4.1 Bisection Thresholding

The operation of eyes detection is done in both detection and tracking mode. Eyes detection is based on a connected component analysis.

The face candidate area is converted into a monochrome smoothed image. Smoothing the area allows to get rid of eventual small gaps in dark areas of the face candidate. The connected component analysis is performed on the smoothed face image as illustrated in Figure 3.

The usual connected component analysis method [9] has been modified here to improve the process efficiency. The eyes are represented on the binary image by two white connected components in the middle of the image (see Figure 3.c). If they are not present then the threshold value is increased until both eyes are detected. This technique is obviously slow, since the threshold value starts from 1 and may go to 255. The approach here uses a bisection threshold method. This means that the threshold value is increased by large steps, for example, by a step of ten. If at a moment eyes were detected, we assume that the best threshold value is within the last step. So the process is now done in a bisection way between last two values. Usually in a normal lighted image, the best threshold value is around 50. Let us examine the case of a best value of 48, and a starting point of 20. In a traditional way, the connected components analysis has to be done 28 times to find the value. In the bisection approach, the steps are as follows: 20, 30, 40 ,50, 45, 47, 48. So we need only 7 steps to find the best threshold value.

4.2 Connected Component Extraction

Another enhancement concerns the connected component analysis itself. The processing is as follows:

1. Proceeding in raster scan order, find the next '1' pixel:

   (a) If only one of its upper and left neighbors has a label, then copy the label (a label identifies a connected component);

   (b) If both have the same label, then copy the label;

   (c) If they have different labels, then copy the upper label, and enter the labels in the equivalence table as equivalent labels;

   (d) Otherwise, assign a new label to the pixel.
2. If there are more pixels to consider, go to step 1.
3. Find the lowest label for each equivalent set in the equivalence table.
4. Scan the picture, replacing each label by the lowest label in its equivalent set.
5. Renumber the equivalence table to remove gaps in the labels.
6. Assign the renumbered labels to components.

In steps 4 and 6, we use a pixel-queue based approach. Since the processed-image usually contains a few connected components with a small size (eyes, and hair on the border), the pixel-queue approach reduces greatly the computing time, since we do not have to scan over and over meaningless pixels. The second improvement concerns the equivalence table mentioned above. Our implementation use a single array (matrix) where operations of labeling and relabeling are done efficiently without any worthless computation.

4.3 Selection of Eyes Candidates

The selection of the appropriate connected components (potential eyes) is done using some heuristics concerning the eyes location on the face:

1) The location of an eye must be inside a certain area of the face (usually in the upper half, not close to borders).

2) The size of an eye must not be too large.

3) The shape must be approximately circular (the fill percentage of the connected component should be more than 60 percent of a rectangular area).

Only two connected components must fulfill the previous criteria in order to be considered as eyes. Then, we apply additional conditions to the pair of potential eyes:

1) The difference between the abscissae of the eyes must not be too small.

2) The difference between the ordinate of the eyes must be small (we can not expect too much head-rotation).

If the potential eyes fulfill all those criteria, it is assumed that they represent the eyes.

In the tracking mode, new position of the eyes is assumed to be near the previous one. We apply this condition to decrease the number of candidates for the eyes.

5 Interpretation of Face Movements

The face movement interpretation is based on analyzing the movements of "Between-The-Eyes" (BTE) point [10]. Each successful location of the BTE is stored, so that it can be used for interpretation and statistics.

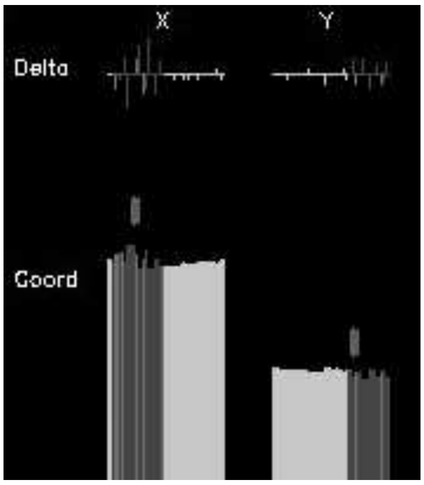

Figure 4: Nodding and head-shaking detection (respectively large bar on top of X-Coord and Y-Coord graphs). Legend: dark gray is extreme state, gray is transient and light gray is stable.

5.1 Basic Moves

Basic moves represent the moves of the head such as: move-up, move-down, move-left, move-right. Those moves are easily detected by a simple difference between two following values of the BTE abscissae for move-left and move-right, and ordinate for move-up and move-down.

5.2 Nodding and Head-Shaking

Nodding (repeated head movements in vertical direction) denotes the left mouse click and head-shaking (repeated head movements in horizontal direction) has a meaning of the right mouse click. We have to distinguish them from repeated move-right and move-left or repeated move-up and move-down.

Nodding or head-shaking motion typically involves two or three cycles. The deviation is plus/minus 0.8 percent of the image size (4-5 pixels at maximum on a 320×240 picture). The deviation also does not last long, typically between 1.0 and 1.4 seconds. The algorithms for nodding and head-shaking detection are identical except that the former processes the abscissae and the latter processes the ordinate.

The head-shaking detection algorithm is as follows. First, each frame is categorized to a stable, transient, or extreme state. Illustration of different states and nodding/head-shaking detection is done in Figure 4, in which each vertical bar represents a frame. The top graphs represent the difference between the two last BTE's x and y coordinates; the bottom graph shows the x and y coordinates explicitly.

In the case of head-shaking detection, if the difference between the maximum value and the minimum value of the abscissae over the last five frames is less that two pixels, then the frame is in a stable state. If in the last 5 frames, the third frame value is the maximum or the minimum then the frame is in an extreme state. Otherwise, the frame is in a transient state. The last frame and the previous one have no state. When the state changes from non-stable to stable, the nodding evaluation process is triggered, so the detection has a two-frame delay. The evaluation is as follows. If there are more than two extreme states between current stable state and the previous stable state, and all adjacent extreme states differ by more than two pixels, then the system assumes that head-shaking has occurred. In the case of move-left or move-right, there are stable states instead of extreme states between or after transient states, so that they can be distinguishable.

6 Cursor Movement Interface

Once the data from the face movements is collected, these data can be used as input to a cursor mapping system. This system aims to move the screen cursor, like with a traditional mouse.

6.1 Cursor Mapping Center

Figure 5.a shows the cursor mapping center (light-gray circle with lines), and the position of the BTE (dark-gray circle with lines).

The cursor mapping center position is determined every time the system switches from detection mode to tracking mode. And if a tracking session is interrupted by a detection stage (the eyes were lost for a few frames, because the head rotated two much, for example), then the cursor mapping center position is kept the same. This interruption can be ignored by modifying the cursor mapping center only if a certain amount of frames were stable before this tracking run. Basically, an interruption of tracking occurs between transient or extreme states. If the head stays in the same position, far from the current cursor mapping center for a long time, then it is assumed that this is a new relaxed position, and the center is updated.

6.2 Cursor displacement

The speed of cursor displacement is proportional to the distance between the cursor mapping center and the BTE. The greater the distance, the faster the cursor movement (Figure 5.b). The center of the user "radar" represents the position of the current cursor mapping center, and the white point represents the current relative distance from the BTE. The point becomes red when in detection mode.

This system has the possibility to simulate "clicks" like with a usual mouse. The nodding movement corresponds to the left click, and the head-shaking corresponds to the right click. The nodding or head-shaking execution time (discussed in section 5.2) can be adjusted to the user capabilities. The nodding and head-shaking used as clicks must not interfere with the mouse displacement. In this system, small head moves within a certain amount of pixels are ignored, and the cursor is not moved. Thus, we have a space where the head can be moved, without moving noticeably the cursor. Since, nodding and head-shaking moves do not require large moves, this space can be used to have a stable click input.

Some other techniques may be used for simulating clicks. One of them is to hold the cursor in the same area for 0.5 sec or so, and then it is considered as a click. If the user needs to click again, the cursor should be moved to a nearby neutral area. This method may be used when the user is

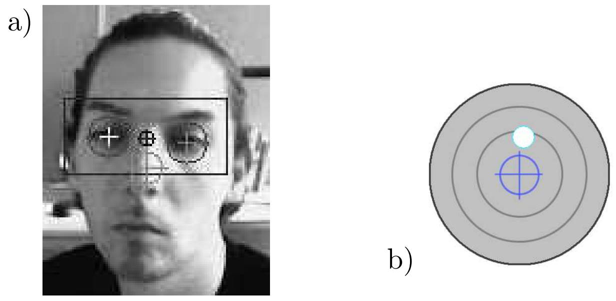

Figure 5: a) The center of cursor mapping on the processed image; b) The relative position of the BTE to the center of mouse mapping (user "radar").

severely disabled, and can not nod or shake the head properly.

7 Experimental Results

The head-controlled interface was implemented on a computer with Pentium 4 3.6 GHz and 2 GB memory. The video input has a 320×240 resolution and a 15 fps frame rate. In full mode (tracking, visualization of the processed image and statistics, and cursor movement control), the system uses 14 percent of the CPU. When all CPU save options are activated, it uses just 6 percent of the processing time. This system was also tested on less powerful machines, and the processing time was still very reasonable. The memory usage is around 20 MB.

The head-controlled user interface is designed to be as unobtrusive as possible, with only a task bar icon, and a little ”radar” window indicating the current distance between the cursor mapping center and the BTE and allowing the user to observe the speed of the cursor and the current mode (detection or tracking).

It is worth noting that an improvement of the frame grabbing rate allows a better control of the cursor, but also induces more CPU consumption. The relation is nearly linear between those factors: double of the frame rate doubles the processing time.

8 Conclusion

In this paper, a head-controlled user interface was presented. This system employs commonly available Web-camera for head movement registration and provides a real-time mouse-like input interface, allowing the user to control the cursor, and to perform left and right clicks without using hands. The face-tracking is entirely marker-free, so the user does not have to wear any extra equipment, making the system totally unobtrusive.

A new approach for connected component analysis was investigated, showing considerable improvements in comparison to the traditional one. A comfortable and easy-to-use cursor mapping interface was created.

The developed head-controlled user interface may be useful in applications like car navigation systems, interfaces for disabled people, games, etc.

References

[1] V. Chauhan, T. Morris, Face and feature tracking for cursor control, *Proc. of SCIA2001: The 12th Scandanavian Conf. on Image Analysis*, 2001.

[2] S. Gurbuzy, K. Kinoshitay, S. Kawato, Real-time human nose bridge tracking in presence of geometry and illumination changes, *Proc. of IWMMS2004: The 2nd Int. Workshop on Man-Machine Symbiotic Systems*, 2004.

[3] A. Kapoor, R. W.Picard, Real-time, fully automatic upper facial feature tracking, *Proc. of 5th Int. Conf. on Automatic Face and Gesture Recognition*, 2002.

[4] H. O. Istance, P. .A Howarth, Input device emulation using an eye-tracker, *Proc. of the Designing Future Interaction Conf.*, Univ. of Warwick, 1994.

[5] J. Gips, P. Olivieri, EagleEyes: An eye control system for persons with disabilities, *Proc. of The 11th Int. Conf. on Technology and Persons with Disabilities*, 1996.

[6] S. Kawato, N. Tetsutani, Real-time detection of between-the-eyes with a circle frequency filter, *Proc. of ACCV2002: The 5th Asian Conf. on Computer Vision*, 2002.

[7] S. Kawato, N. Tetsutani, Circle-frequency filter and its application, *Proc. of IWAIT2001: Int. Workshop on Advanced Image Technology*, pp.217-222, 2001.

[8] S. Kawato, J. Ohya, Automatic skin-color distribution extraction for face detection and tracking, *Proc. of ICSP2000: The 5th Int. Conf. on Signal Processing*, vol.II, pp.1415-1418, 2000.

[9] R. Stiefelhagen, J. Yang, A. Waibel, Tracking eyes and monitoring eye gaze, *Proc. of the Workshop on Perceptual User Interfaces*, pp. 98-100, 1997.

[10] S. Kawato, J. Ohya, Real-time detection of nodding and head-shaking by directly detecting and tracking the Between-Eyes, *Proc. of FG2000: The 4th Int. Conf. on Automatic Face and Gesture Reconition*, pp.40-45, 2000.