Bare-Hand Human-Computer Interaction

Christian von Hardenberg
Institut für Elektronik
Technische Universität Berlin
Einsteinufer 17, 10587 Berlin, Germany
+49-30-7889 5905
kiwahiga@linux.zrz.tu-berlin.de

François Bérard
CLIPS-IMAG
BP 53, 38041 Grenoble Cedex 9
France
(+33)476 514 365
francois.berard@imag.fr

ABSTRACT

In this paper, we describe techniques for barehanded interaction between human and computer. Barehanded means that no device and no wires are attached to the user, who controls the computer directly with the movements of his/her hand.

Our approach is centered on the needs of the user. We therefore define requirements for real-time barehanded interaction, derived from application scenarios and usability considerations. Based on those requirements a finger-finding and hand-posture recognition algorithm is developed and evaluated.

To demonstrate the strength of the algorithm, we build three sample applications. Finger tracking and hand posture recognition are used to paint virtually onto the wall, to control a presentation with hand postures, and to move virtual items on the wall during a brainstorming session. We conclude the paper with user tests, which were conducted to prove the usability of bare-hand human computer interaction.

Categories and Subject Descriptors

H.5.2 **[Information interfaces and presentation]**: User interfaces -*Input devices and strategies*; I.5.5 **[Pattern recognition]**: Implementation - *Interactive systems*.

General Terms

Algorithms, Design, Experimentation, Human Factors

Keywords

Computer Vision, Human-computer Interaction, Real-time, Finger Tracking, Hand-posture Recognition, Bare-hand Control.

1. INTRODUCTION

For a long time research on human-computer interaction has been restricted to techniques based on the use of a graphic display, a keyboard and a mouse. Recently this paradigm has changed. Techniques such as vision, sound, speech recognition, projective displays and context-aware devices allow for a much richer, multi-modal interaction between man and machine.

Today there are many different devices available for hand-based human-computer interaction. Some examples are keyboard, mouse, track-ball, track-pad, joystick, electronic pens and remote controls. More sophisticated examples include cyber-gloves, 3D-mice (e.g. Labtec's Spaceball) and magnetic tracking devices (e.g. Polhemus' Isotrack). But despite the variety of new devices, human-computer interaction still differs in many ways from human-to-human interaction. Natural interaction between humans does not involve devices because we have the ability to sense our environment with eyes and ears. In principle, the computer should be able to imitate those abilities with cameras and microphones.

In this paper, we will take a closer look at human-computer interaction with the bare hand. In this context, "bare" means that no device has to be in contact with the body to interact with the computer. The position of the hand and the fingers will be used to control applications directly.

Our approach will be centered on the needs of the user. Requirements derived from usability consideration will guide our implementation, i.e. we will not try to solve general computer vision problems, but rather find specific solutions for a specific scenario.

In the next section of the paper, we will describe applications, where bare-hand input is superior to traditional input devices. Based on these application scenarios, we will set up functional and non-functional requirements for bare-hand human-computer interaction systems.

In the fourth section, there will be an overview about possible approaches and related work to the problem. Because none of the current hand-finding and -tracking systems sufficiently fulfill our requirements, we developed our own algorithm, which will be described in sections five and six. To demonstrate the strength of the algorithm, we build three sample applications that are presented in section seven. We conclude the paper with an evaluation of the performance of our algorithm and describe user tests, which were conducted to prove the usability of bare-hand human-computer interaction.

2. APPLICATIONS

We found three main application scenarios for bare-hand human-computer interaction. First, in many cases the bare hand is more practical than traditional input devices:

- During a presentation, the presenter does not have to move back and forth between computer and screen to select the next slide.
- Remote controls for television sets, stereos and room lights could be replaced with the bare hand.
- During video conferences, the camera's attention could be acquired by stretching out a finger, similar to a classroom situation.
- Household robots could be controlled with hand gestures.
- Mobile devices with very limited space for user interfaces could be operated with hand gestures.

Additionally, perceptual interfaces allow the creation of computers that are not perceived as such. Without monitor, mouse and keyboard, a computer can hide in many places, such as household appliances, cars, vending machines and toys. The main advantage of perceptual interfaces over traditional buttons and switches are as follows:

- Systems can be integrated on very small surfaces.
- Systems can be operated from a certain distance.
- The number of mechanical parts within a system can be reduced, making it more durable.
- Very sleek designs are possible (imagine a CD-Player without a single button).
- Systems can be protected from vandalism by creating a safety margin between the user and the device.
- In combination with speech recognition, the interaction between human and machine can be greatly simplified.

Finally, there is a class of applications, which can be built in combination with a projector. Virtual objects that are projected onto the wall or onto a table can be directly manipulated with the fingers. This setup can be useful in several ways:

- Several persons can simultaneously work with the objects projected onto the wall.
- Physical systems, such as a schedule on the wall, can be replaced with digital counterparts. The digital version can be easily stored, printed, and sent over the Internet.
- If projector and camera are mounted in a place that is not accessible for the user, an almost indestructible interface can be built. To the user, the computer physically only consists of the wall at which the interface is projected.

It has to be noted, that speech recognition systems might also be able to provide some of the listed properties. Vision-based techniques have the advantage that they do not disturb the flow of conversation (e.g. during a presentation) and work well in noisy environments (e.g. for public-space installations).

3. REQUIREMENTS

In this section, we will define requirements for real-time human-computer interaction with bare hands that will guide our implementation and evaluation later on.

3.1 Functional Requirements

Functional requirements can be described as the collection of services that are expected from a system. For a software system, these services can cover several layers of abstraction. In our context, only the basic services are of interest.

We identify three essential services for vision-based human-computer interaction: detection, identification and tracking. We will briefly present the three services and describe how they are used by our envisaged applications.

3.1.1 Detection

Detection determines the presence and position of a class of objects in the scene. A class of objects could be body parts in general, faces, hands or fingers. If the class contains only one object type, and there is just one object present in the scene at a time, detection suffices to build simple applications.

For example, if we detect the presence of fingertips and we constrain our application to one fingertip at a time, the detection output can be used to directly control a mouse pointer position. For more complex applications, such as hand posture recognition and multi-handed interaction, we will need an additional identification and tracking stage.

3.1.2 Identification

The goal of identification is to decide which object from a given class of objects is present in the scene.

For bare-hand interaction, different identification tasks are potentially interesting:

- Identification of a certain hand posture: Many applications can be realized with the reliable identification of a stretched out forefinger. Examples: Finger-driven mouse pointer, recognition of space-time gestures, moving projected objects on a wall, etc.
- Number of fingers visible: Applications often need only a limited number of commands (e.g. simulation of mouse buttons, “next slide”/”previous slide” command during presentation). Those commands can be controlled by the number of fingers presented to the camera.
- 2D-positions of fingertips and the palm: In combination with some constraints derived from the hand geometry, it is possible to decide which fingers are presented to the camera. Theoretically, thirty-two different finger configurations can be detected with this information. For non-piano players only a subset of about 13 postures will be easy to use, though.
- 3D-position of all fingertips and two points on the palm: As shown by [7], those parameters uniquely define a hand pose. Therefore, they can be used to extract complicated postures and gestures. An important application is automatic recognition of hand sign languages.

All those identification tasks are solved in the literature, as long as the background is uniform and the speed of hand movement is restricted (see section four). In order to limit the difficulty of the problem, we limit ourselves to the first two points. This allows us to built a system with a minimum number of constraints on the user and the environment while still providing the required services for our targeted applications.

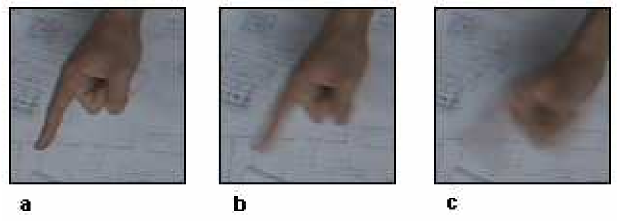

Figure 1. Motion blurring. (a) Resting hand (b) Normal movement (c) Fast Movement.

3.1.3 Tracking

In most cases, the identified objects will not rest in the same position over time. If two objects of the same class are moving in the scene, tracking is required to be able to tell which object moved where between two frames.

There are two basic approaches to deal with the tracking problem. First, it is possible to remember the last known position of an identified object. Given some known constraints about the possible movements of an object between two frames, a tracking algorithm can try to follow the object over time.

The second possibility is to re-run the identification stage for each frame. While this approach might seem rather crude and also requires a very fast identification algorithm, it might be the only feasible tracking technique for unconstrained hand motion for two reasons:

- Measurements show that hands can reach speeds of 5 m/s during normal interaction. At a frame rate of 25 frames/s the hand "jumps" with steps of up to 20cm per frame.
- As shown in Figure 1, fast finger motion results in strong motion burring. The blurred fingers are almost impossible to identify and therefore lead to unknown finger positions during fast motion.

The two effects combined typically result in distances of about one meter between two identified finger positions, for fast hand movements. This requires the tracking algorithm to search through most of the image for each frame. By skipping the tracking stage altogether one does not loose much processing speed, but gains a lot of stability[1].

3.2 Non-Functional Requirements

Non-functional requirements describe the minimum quality expected from a service.

3.2.1 Latency

Latency is the time gap between an action of the user and the system response. There is no system without latency, so the basic question is what is the maximum acceptable latency for our system.

Several studies ([10], [18]) have shown that user performances degrade significantly at high latencies. However, it is difficult to derive a maximum acceptable lag from those studies because the answer differs, depending on the chosen task and the performance degradation is gradual.

We therefore take a different approach to the problem: the described applications require real-time interaction, which we define as interaction without a perceivable delay. A classical experiment conducted by Michotte and reported in [3] shows that users perceive two events as connected by "immediate causality", if the delay between the events is less than 50ms. The perception of immediate causality implies that the user does not notice a delay between the two events. We therefore conclude that the minimum acceptable latency for real-time applications is in the area of 50ms, resulting in a required minimum processing frequency of 20Hz.

3.2.2 Resolution

The required spatial resolution depends on the application. For point-and-click tasks, the smallest possible pointer movement should be at most as large as the smallest selectable object on the screen. For other applications, such as simple gesture interfaces, the output resolution does not affect the quality of the service, but detection and identification processes require a minimum input resolution. For example, we found it difficult to identify fingers with a width below six pixels in the image.

3.2.3 Stability

A tracking method can be called stable if the measured position does not change, as long as the tracked object does not move. There are several possible sources of instability, such as changing light conditions, motion of distracting objects and electrical noise.

The stability of a system can be measured by calculating the standard deviation of the output data for a non-moving object over a short period.

As a necessary condition, the standard deviation has to be smaller than the smallest object the user can select on a screen (e.g. a button). As a sufficient condition, it should be smaller than the smallest displayable position change of the pointer to avoid annoying oscillation of the pointer on the screen.

4. RELATED WORK

In the last ten years, there has been a lot of research on vision-based hand gesture recognition and finger tracking. Interestingly there are many different approaches to this problem with no single dominating method. The basic techniques include color segmentation [8], [12], infrared segmentation [13], blob-models [6], contours [13], [9], [15], correlation [4], [11] and wavelets [17].

Typical sample applications are finger driven mice [11], [12], finger driven drawing applications [4], [6], [9], bare-hand game control [5], [15], and bare-hand television control [5].

Most authors use some kind of restriction, to simplify the computer vision process:

- Non real-time calculations [17]
- Colored gloves [8]
- Expensive hardware requirements (e.g. 3D-camera or infrared-camera) [13], [14]
- Restrictive background conditions [15]

[1] Every tracker looses track from time to time (e.g. due to occlusion) and has to be restarted. By "restarting" the system at each frame, many tracking problems can be avoided.

- Explicit setup stage before starting the tracking [4]
- Restrictions on the maximum speed of hand movements [4], [6], [9], [11]

Most systems additionally have problems in the case of changing light conditions and background clutter. The systems described in [6] and [9] seem to perform the best under such difficult conditions.

None of the presented work provides a robust tracking technique for rapid hand movements. In addition, most systems require some kind of setup-stage before the interaction can start. Finally, none of the reviewed systems allows simultaneous tracking of several fingers in real-time.

Because we believe that those three points are crucial for most bare-hand human-computer interaction applications, we decided to develop an own finger-finding algorithm, which will be described in the next two sections.

5. HAND SEGMENTATION

When processing video images, the basic problem lies in the extraction of information from vast amount of data. The Matrox Meteor frame grabber, for example, captures over 33 megabytes of data per second, which has to be reduced to a simple fingertip position value in fractions of a second.

The goal of the segmentation stage is to decrease the amount of image information by selecting areas of interest. Due to processing power constraints, only the most basic calculations are possible during segmentation. Typical hand segmentation techniques are based on stereo information, color, contour detection, connected component analysis and image differencing.

Each technique has its specific disadvantages:

Stereo image based segmentation requires a hardware setup that currently only can be found in laboratories.

Color segmentation is sensitive to changes in the overall illumination [19]. In addition, it is prone to segmentation errors caused by objects with similar colors in the image. It also fails, if colors are projected onto the hand (e.g. during a presentation).

Contour detection tends to be unreliable for cluttered backgrounds. Much stability is obtained by using a contour model and post-processing with the condensation algorithm, but this restricts the maximum speed of hand movement [2].

Connected component algorithms, tend to be heavy in computational requirements, making it impossible to search through the whole image in real-time. Successful systems employ tracking techniques, which again restrict the maximum speed of movement [6].

Image differencing generally only works well for moving objects and requires sufficient contrast between foreground and background. The next sub-section will show how it nevertheless can work quite reliably with the help of simple heuristics.

Looking at the failure-modes of the different segmentation techniques, the obvious idea is to combine several techniques to get results that are more robust.

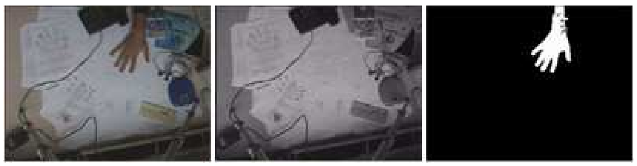

Figure 2. Image differencing with reference image. From left to right: input image, reference image, output image.

Surprisingly, a quantitative comparison of low-level image processing techniques[2] showed that all evaluated methods tend to fail under similar conditions (fast hand motion, cluttered background). For this reason, a combination of techniques does not yield a much better performance.

After comparing the different possible techniques qualitatively as well as quantitatively, we decided to work with a modified image differencing algorithm, which will be described in the next subsection.

5.1 Smart Image Differencing

Studies on human perception show that the visual system uses changes in luminosity in many cases to set the focus of attention. A change of brightness in one part of the visual field, such as a flashing light, attracts our attention. Image differencing follows the same principle. It tries to segment a moving foreground from a static background by comparing the gray-values of successive frames.

Inter-frame image differencing is only useful if the foreground object constantly moves. If we take the difference between the actual image and a *reference image* instead, it is also possible to find resting hands (see Figure 2). To cope with changes in illumination the reference image is constantly updated with newly arriving image using the following formula [16]:

$$\forall x, \forall y \quad R_t(x, y) = \frac{N-1}{N} \cdot R_{t-1}(x, y) + \frac{1}{N} \cdot I(x, y) \qquad (1)$$

With *R* standing for the reference image and *I* for the newly arrived frame. The formula calculates a running average over all frames, with a weighting that decreases exponentially over time. For setups in front of a wall or white board, we found that the user practically never rests with his hand in the same position for more than 10 seconds, which implies a value of about 500 for *N*.

The main problem with this type of reference image updating is that dark, non-moving objects, such as the body are added to the reference image. If the user moves his/her hand into regions with dark objects in the reference image, there is not sufficient contrast between foreground and background to detect a difference.

We found a simple but effective solution to this problem. In our setup, the background is generally lighter than the foreground (white walls, white boards). For this reason, we can update dark

[2] Segmentation based on color, image differencing, connected components and correlation was applied to a set of 14 different hand labeled sequences

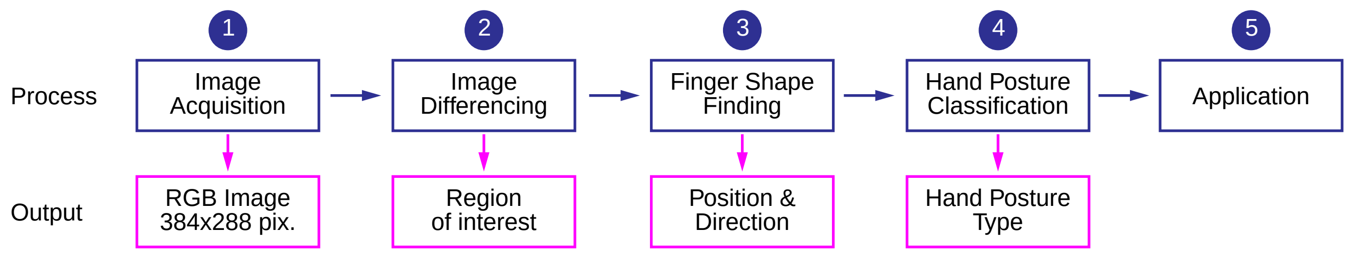

Figure 3. The finger-finding and hand-posture recognition process.

regions slowly, but light regions instantly. The value N in formula (1) is calculated as follows:

$$\forall x, \forall y \quad N(x,y) = \begin{cases} 1 & for\ I_{t-1}(x,y) - I_t(x,y) \le 0 \\ \approx 500 & for\ I_{t-1}(x,y) - I_t(x,y) > 0 \end{cases} \qquad (2)$$

With this modification, the algorithm provides the maximum contrast between foreground and background at any time.

6. FINDING FINGERS AND HANDS

Figure 3 gives a schematic overview of the complete finger-finding and hand-posture recognition process. The system searches for fingertips first and uses the found positions to derive higher-level knowledge, such as hand postures or gestures.

6.1 Fingertip Shape Finding

This section will present a simple, fast and reliable algorithm that finds both the position of fingertips and the direction of the fingers, given a fairly clean segmented region of interest.

Figure 4 shows some typical finger shapes extracted by the image-differencing process. Looking at these images, one can see two overall properties of a fingertip:

- The center of the fingertips is surrounded by a circle of filled pixels. The diameter of the circle is defined by the finger width.
- Along a square outside the inner circle, fingertips are surrounded by a long chain of non-filled pixels and a shorter chain of filled pixels (see Figure 5).

To build an algorithm that searches these two features, several parameters have to be derived first:

Diameter of the little finger (d1): This value usually lies between 5 and 10 pixels and can be calculated from the distance between the camera and the hand.

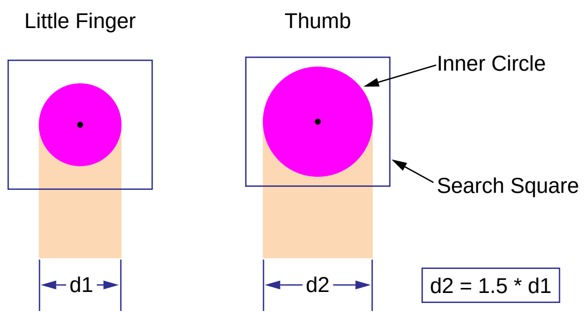

Figure 5. A simple model of the fingertip.

```
∀(x, y) ∈  Region_of_Interest
  Add number of filled pixels in circle with
  diameter d1 and center (x, y)

  If (filled_pixel_nb < circle_area)          (1)
     Continue loop

  Calculate number of filled pixels along
  search square with diameter d3

  If (filled_pixel_nb < min_pixel) or
     (filled_pixel_nb > max_pixel)            (2)
     Continue loop

  If (connected_filled_pixel_nb <
     filled_pixel_nb – error_margin)          (3)
     Continue loop

  Memorize (x, y) position
```

Figure 6. Fingertip-finding algorithm

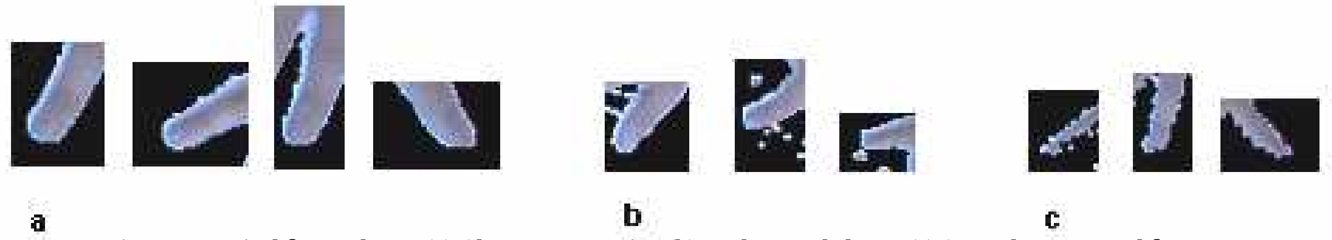

Figure 4. Typical finger shapes (a) Clean segmentation (b) Background clutter (c) Sparsely segmented fingers.

Diameter of the thumb (d2): Experiments show that the diameter is about 1.5 times the size of the diameter of the little finger.

Size of the search square (d3): The square has to be at least two pixels wider than the diameter of the thumb.

Minimum number of filled pixels along the search square (min_pixel): As shown in Figure 5 the minimum number equals the width of the little finger.

Maximum number of filled pixels along the search square (max_pixel): Geometric considerations show that this value is twice the width of the thumb.

Given those parameters, we wrote the straightforward fingertip finding algorithm reported in Figure 6.

The algorithm performs three checks to find out whether a given position (x, y) is a fingertip.

(1) There has to be a sufficient number of filled pixels around the close neighborhood of the position (x, y).
(2) There has to be the correct number of filled and un-filled pixels along the described square around (x, y).
(3) The filled pixels along the square have to be connected in one chain.

This basic algorithm runs easily at real-time and reliably finds possible fingertips. We implemented two enhancements to further improve the stability. First, it is useful to define a minimum distance between two fingertips to avoid classifying two pixels next to each other as different fingertips. Second, the middle position of the chain of inner pixels along the search square shows the direction of the finger. This information can be used to determine whether the found fingertip is connected to an outstretched finger.

6.2 Hand posture Classification

The second part of the algorithm analyzes the relationship between the found fingers.

As a first step, a standard connected component analysis algorithm is used to analyze which of the found fingers belong to the same hand. As a by-product, the size of the hand-region is calculated. This can be used to filter out small finger-shaped objects, such as pens.

In a next step, the fingers are sorted into the right geometric order (minimum distance between every pair). Afterwards the directions and positions of fingers relative to each other allow calculating an approximation for the center of the palm. Fingers can then be classified by their position relative to the palm and their position to each other.

6.3 Evaluation

In section three we set up requirements for real-time bare-hand human computer interaction. While the functional requirements have to be evaluated on a system level, we can already check at this point if the non-functional requirements are met.

Our measurements were done one a Pentium III 1000MHz machine with 384x288 sized images. To test accuracy and robustness we ran the finger finding algorithm on 12 sequences with varying light conditions (daylight, neon light, diffuse and with shadows), different degrees of background clutter and different speeds of movement. The output of the finger-tracker was compared to hand-labeled ground truth data to calculate mean error and error variance.

Table 1. Dropped and misclassified frames (out of 25 frames)

Motion	Back-ground	Light Condition	Dropped Frames	Misclas. Frames
Slow (<2.5 m/s)	White	Diffuse Daylight	0	0
		Daylight with Shadows	0	1
		Diffuse Neon light	0	0
	Clutter	Diffuse Daylight	0	0
		Daylight with Shadows	0	0
		Diffuse Neon light	4	0
Fast (<4.5 m/s)	White	Diffuse Daylight	3	3
		Daylight with Shadows	5	2
		Diffuse Neon light	5	0
	Clutter	Diffuse Daylight	7	0
		Daylight with Shadows	1	0
		Diffuse Neon light	8	0

6.3.1 Latency

The total latency of the algorithm was between 26 and 34ms, depending on the number of pixels in the region of interest. Image differencing alone takes about 10ms. The maximum latency is therefore still well below the required maximum latency of 50ms. We intentionally left some room for the latency of image acquisition and graphical output.

6.3.2 Robustness

For each sequence two types of frames have been counted:

- Dropped frames: Frames in which no finger could be found
- Misclassified frames: Frames in which the finger position was off by more than 10 pixels from the right position, or frames in which the nearby finger-shadow has been tracked instead of the finger itself.

Dropped frames usually occur if the finger moves very fast. In this case they do not cause problems to most applications, because the resting position of a finger is usually much more important than the fast moving position. Misclassified frames, on the other hand, are quite annoying. If the finger controls a pointer, for example, this pointer might jump forth and back erratically between correctly and misclassified finger positions.

Table 1 shows that the algorithm is quite robust for most circum-stances. The occasional misclassified frames can be explained with nearby shadows and body parts that resemble a finger. a simple stabilization step, that always chooses the finger-position closest to the last known position will eliminate most of those problems later on.

6.3.3 Accuracy

The accuracy was calculated for all correctly classified frames. The

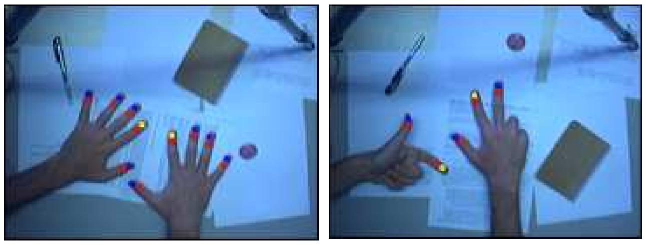
Figure 7. Finger classification

mean error for the evaluate sequences was between 0.5 and 1.9 pixels with variances between 0.1 and 2.0. As expected, fast finger movements are tracked with lower accuracy than slow movements, due to motion blurring. Taking into account the error margin of the hand labeling process, the overall accuracy of the algorithm is better than one pixel and therefore sufficient for precise point-and-click tasks.

6.3.4 Finger Classification

Figure 7 shows two examples of the finger classification algorithm. The finger positions (blue dots) and directions (red dots) are reliably found for all fingers in front of a cluttered background. Other finger-like objects such as the pen or the ball are ignored. Additionally, the forefingers (yellow dots) of the two hands are correctly found. The ten fingers are grouped into two different hand objects (not visible in the picture)[3].

7. SAMPLE APPLICATIONS

We developed three applications, named *FingerMouse*, *FreeHandPresent* and *BrainStorm* for this paper. All of them aim to improve the interaction between human and computer for a specific scenario, and all demonstrate different capabilities of the finger-finding and hand-posture recognition system.

7.1 Description of the applications

7.1.1 FingerMouse

The *FingerMouse* system allows control of the mouse pointer with the bare hand. The user just moves an outstretched forefinger in front of the camera, to position the mouse pointer on the screen. Mouse-clicks are generated by keeping the finger in the same position for one second. The mouse-wheel is activated by stretching out all five fingers. In combination with a projector, the system can be used to control Windows applications, such as the *Internet Explorer* or *Paint*, directly on a wall (see Figure 8a and b). The finger replaces both a physical mouse as well as a mouse pointer, allowing for fast and intuitive interaction.

7.1.2 FreeHandPresent

The second system is built to demonstrate how simple hand postures can be used to control an application. A typical scenario where the user needs to control the computer from a certain distance is during a presentation. The *FreeHandPresent* system

[3] Of course the two demonstrated cases are just examples for many different possible conditions and hand states. MPEG movies of the finger finder and of all described applications can be viewed at http://iihm.imag.fr/hardenbe/Videos.htm

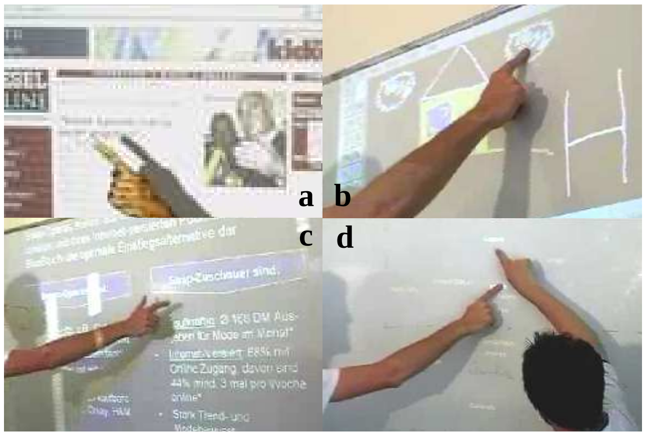

Figure 8. Bare-hand human-computer interaction.
(a) Finger controlled web browser (b) Painting with the finger
(c) Controlling a presentation with hand postures
(d) Multi-user spatial reorganization of text items.

makes it possible to replace existing remote-control solutions with the bare hand. It uses the following hand postures to control a presentation: two outstretched fingers for "next slide" (see Figure 8c), three outstretched finger for "previous slide" and five outstretched fingers to open a "slide menu". The slide menu makes it possible to jump to a specific slide quickly during a presentation, by selecting one of the displayed side numbers with the finger.

7.1.3 Brainstorm

The last system was built to demonstrate multi-user/multi-hand tracking capabilities of our system. The application scenario is a brainstorming session. Normally such sessions consist of two phases: first, a large number of ideas are collected from the participants and pinned to the wall. Second, the items on the wall are sorted and categorized.

With the BrainStorm system, users can type their ideas "to the wall", using a wireless keyboard. In the second phase, everyone can walk up to the wall and rearrange items with his/her fingers. An item is selected by resting on it with an outstretched finger for half a second. Selected items can be moved freely on the wall. To unselect an item, the user again makes a short pause with the finger. Several users can rearrange items in parallel (see Figure 8d). The main advantage of the virtual brain storming system is that results can be saved, printed and send over the Internet at any time.

7.2 Evaluation of the applications

All applications fulfilled the functional requirements defined in section three. They worked in real-time (20-25Hz) and proved to be sufficiently precise and robust to fulfill the chosen tasks. Inexperienced users were able to use all of the applications after a short explanation. Especially the selecting/clicking technique with a short pause of the finger proved to be very intuitive.

Nevertheless, we noticed one main problem during our evaluation: the projected background usually changes a lot during interaction. The image-differencing layer therefore produces plenty of "false"

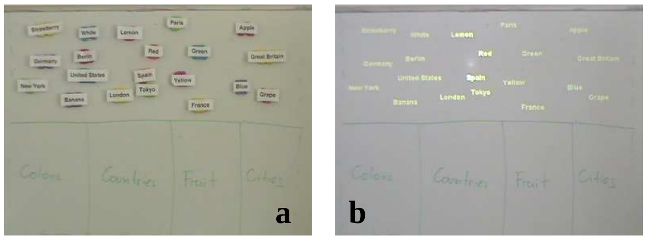

Figure 9. User study (a) Physical objects (b) Virtual objects.

foreground objects, which might be accidentally classified as fingers. There are two ways to cope with this problem.

- For applications such as *FreeHandPresent* the user can do his/her hand sign at the side or above the presentation in a pre-defined “control area”.
- In all other cases, it is possible to eliminate the disturbing effect of the projection by illumination the room (e.g. sun-light or a strong lamp).

In principle, this problem could also be solved, with synchronized camera-projector setups, which capture images during short periods of blacked out projection.

To prove the usability of barehanded human-computer interaction more formally, we arranged a user study with the *BrainStorm* system. Eighteen inexperienced users had to group twenty projected words on the wall into four categories (cities, countries, colors and fruit). The same experiment was done with physical objects (words glued to magnets, see Figure 9). Half of the users did the physical object-sorting first, the other half started with the virtual items.

On average, it took users 37 seconds to sort the physical objects and 72 seconds for the virtual objects, resulting in a 95% increase in time. The difference can be mainly explained with the selection and un-selection pause of 0.5 seconds, which adds up to 20 seconds for the 20 items on the wall.

8. CONCLUSION

In this paper, we described how a computer can be controlled with the bare hand. We developed a simple but effective finger finding algorithm that runs in real-time at a wide range of light conditions. Other than in previous work, our system does not constrain the hand movement of the user. Also, there is no set-up stage. Any user can simply walk up to the wall and start interacting with the system.

The described user tests show that the organization of projected items on the wall can be easily accomplished with bare hand interaction. Even though the system takes more time than its physical counterpart, we think that it is still very useful: many value-adding services, such as printing and storing, can only be realized with the virtual representation.

Further research will be necessary to find a faster selection-mechanism and to improve the segmentation with a projected background under difficult light conditions.

9. REFERENCES

[1] Bérard, F. Vision par ordinateur pour l’interaction homme-machine fortement couplée, Doctoral Theses, Université Joseph Fourier, Grenoble, 1999.

[2] Blake, A., Isard, M., and Reynard, D. Learning to track the visual motion of contours, Artificial Intelligence, 78, 101-134, 1995.

[3] Card, S. Moran, T. Newell, A. The Psychology of Human-Computer Interaction, Lawrence Erlbaum Associates, 1983.

[4] Crowley, J., Bérard, F., and Coutaz, J. Finger tacking as an input device for augmented reality, Automatic Face and Gesture Recognition, Zürich, 195-200, 1995.

[5] Freeman, W., Anderson, D. and Beardsley, P. Computer Vision for Interactive Computer Graphics, IEEE Computer Graphics and Applications, 42-53, Mai-June 1998.

[6] Laptev, I. and Lindeberg, T. Tracking of Multi-State Hand Models Using Particle Filtering and a Hierarchy of Multi-Scale Image Features, Technical report ISRN KTH/NA/P-00/12-SE, September 2000.

[7] Lee, J. and Kunii, T. Constraint-based hand animation, in Models and techniques in computer animation, 110-127, Springer Verlag, Tokyo, 1993.

[8] Lien, C. and Huang, C. Model-Based Articulated Hand Motion Tracking For Gesture Recognition, Image and Vision Computing, vol. 16, no. 2, 121-134, February 1998.

[9] MacCormick, J.M. and Isard, M. Partitioned sampling, articulated objects, and interface-quality hand tracking, European Conference on Computer Vision, Dublin, 2000.

[10] MacKenzie, I. and Ware, C. Lag as a determinant of Human Performance in Interactive Systems. Conference on Human Factors in Computing Systems, 488-493, New York, 1993.

[11] O'Hagan, R. and Zelinsky, A. Finger Track - A Robust and Real-Time Gesture Interface, Australian Joint Conference on Artificial Intelligence, Perth, 1997.

[12] Quek, F., Mysliwiec, T. and Zhao, M. Finger mouse: A freehand pointing interface, International Workshop on Automatic Face- and Gesture-Recognition, Zürich, 1995.

[13] Rehg, J. and Kanade, T. Digiteyes: Vision-based human hand tracking, Technical Report CMU-CS-93-220, School of Computer Science, Carnegie Mellon University, 1993.

[14] Sato, Y., Kobayashi, Y. and Koike, H. Fast Tracking of Hands and Fingertips in Infrared Images for Augmented Desk Interface, International Conference on Automatic Face and Gesture Recognition, Grenoble, 2000.

[15] Segen, J. GestureVR: Vision-Based 3D Hand Interface for Spatial Interaction, ACM Multimedia Conference, Bristol, 1998.

[16] Stafford-Fraser, J. Video-Augmented Environments, PhD theses, Gonville & Caius College, University of Cambridge, 1996.

[17] Triesch, J. and Malsburg, C. Robust Classification of Hand Postures Against Complex Background, International Conference On Automatic Face and Gesture Recognition, Killington, 1996.

[18] Ware, C. and Balakrishnan, R. Researching for Objects in VR Displays: Lag and Frame Rate, ACM Transactions on Computer-Human Interaction, vol. 1, no. 4, 331-356, 1994.

[19] Zhu, X., Yang, J. and Waibel, A. Segmenting Hands of Arbitrary Color, International Conference on Automatic Face and Gesture Recognition, Grenoble, 2000.