Architecture and applications of the FingerMouse
A smart stereo camera for wearable computing HCI

Pers Ubiquit Comput (2008) 12:97–110
DOI 10.1007/s00779-006-0109-0

ORIGINAL ARTICLE

Architecture and applications of the FingerMouse: a smart stereo camera for wearable computing HCI

Patrick de la Hamette · Gerhard Tröster

Received: 17 May 2006 / Accepted: 21 July 2006 / Published online: 9 January 2007

Abstract In this paper we present a visual input HCI system for wearable computers, the FingerMouse. It is a fully integrated stereo camera and vision processing system, with a specifically designed ASIC performing stereo block matching at 5 Mpixel/s (e.g. QVGA 320×240 at 30 fps) and a disparity range of 47, consuming 187 mW (78 mW in the ASIC). It is button-sized (43 mm $\times$ 18 mm) and can be worn on the body, capturing the user's hand and processing in real-time its coordinates as well as a 1-bit image of the hand segmented from the background. Alternatively, the system serves as a smart depth camera, delivering foreground segmentation and tracking, depth maps and standard images, with a processing latency smaller than 1 ms. This paper describes the FingerMouse functionality and its applications, and how the specific architecture outperforms other systems in size, latency and power consumption.

Keywords Wearable computing · Stereo vision · Mobile embedded vision · Hand tracking · Foreground segmentation · HCI

1 Introduction

As a new generation of computers, wearable computers are worn on the user's body or are even integrated in his textiles. This allows for new application scenarios: the computer becomes a digital assistant helping the user perform certain tasks. The system shall not obstruct the user in any way, his hands should be free. Ideally the digital assistant provides useful information (e.g. via a head-up display, or sound) without requiring explicit user interaction. In many situations though, user input to the system will be necessary. Obviously the classic input devices like a keyboard or a mouse do not fit into the wearable computing scenario. A new class of human–computer interaction (HCI) devices is required.

This paper describes such a new wearable device, the FingerMouse. Its primary use is to capture the movement and shape of the user's hand when moving in front of the device's cameras, enabling him to interact with the wearable computer using his bare hands (hence the name FingerMouse). This, and further applications scenarios are described in Sect. 6.

The FingerMouse uses two cameras and its own processing power to run vision algorithms, to capture a scene and transmit the following outputs:

1. Hand shape: the hand shape is computed as a 1-bit bitmap. This bitmap defines the segmentation of pixels in the scene into either hand/foreground or background.
   In a more general sense, the FingerMouse is a binary depth sensing camera, classifying pixels into foreground, when they belong to objects with a distance $Z \in [Z_{\min}, Z_{\max}]$. $Z_{\min}$ and $Z_{\max}$ can be configured to values from Z_{prox} to $+\infty$. Objects closer than the maximal proximity Z_{prox} produce noise. All other pixels are set to the background class (drawn in black in Fig. 1). The FingerMouse

P. de la Hamette (✉) · G. Tröster
Wearable Computing Lab, ETH Zurich, Switzerland
e-mail: pdelaham@ife.ee.ethz.ch
URL: http://www.wearable.ethz.ch

G. Tröster
e-mail: troester@ife.ee.ethz.ch

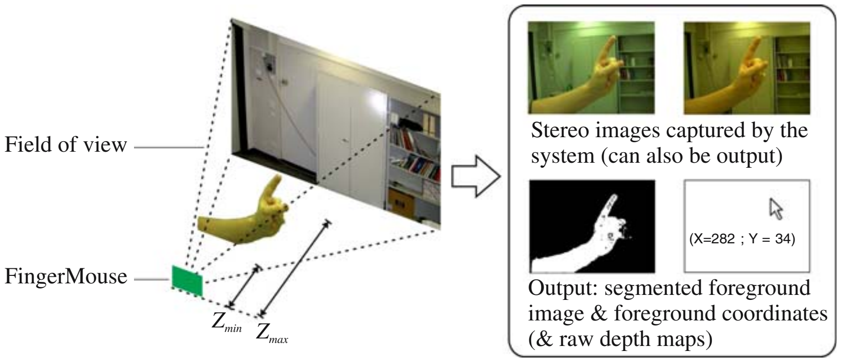

Field of view
FingerMouse
Z_{min}
Z_{max}
Stereo images captured by the system (can also be output)
(X=282 ; Y = 34)
Output: segmented foreground image & foreground coordinates (& raw depth maps)

Fig. 1 FingerMouse functionality: scene capture, outputs

transmits this image to a (wearable) computer, acting as a segmenting smart camera. On the computer, a higher level algorithm could use this images for further processing (c.f. Sect. 6).

2. Hand movement: the hand position (absolute coordinates) in the captured images is computed. This allows the user to control an *X*–*Y* pointer, similar as a PC-mouse does. For more simple interactions, hand movements (up, down, etc.) are also useful, since they do not require visual feedback as a pointer does.
3. Scene depth image (disparity map): instead of the segmented output, the FingerMouse can also deliver a measurement of the depth of all the pixels. The depth information is the raw disparity measurement from a stereo block matching algorithm (SAD result, L $\rightarrow$ R or R $\rightarrow$ L), described in Sect. 2.
4. Standard images: the images from the system's two cameras can also be read. This allows to combine the foreground segmentation result (“hand shape” of the scene with standard camera images yielding the same perspective.

The following features characterize the FingerMouse:

1. Full integration: the embedded system includes all the parts used for image capturing, image processing and power regulation off a battery voltage. It computes the results mentioned above autonomously and transmits them to a (wearable) computer.
2. Real-time operation: as a HCI device, the computation has to be done online. The FingerMouse operates in real-time, processing several frames per second.
3. Low latency/high usability: when using the system with visual feedback (e.g. *X*–*Y* pointer on a screen), the user and the machine become a closed-loop system. The smaller the tracking latency (or lag) is, the higher the usability. HCI research suggests that values under 50 ms are acceptable and that latency is more important to usability than the measurement accuracy [1, 2].
4. No calibration: the device immediately works after power-up. This allows it to be turned on shortly (e.g. when user-interaction is needed), and it increases usability because it does not interrupt the user's work flow.
5. High computation performance: the FingerMouse processes images with a throughput of 5–6 Mpixel/s, e.g. 320 $\times$ 240 images at 30–37 fps. We developed a specific ASIC for the image computation, which computes at 20–25 G Operations per second (at 80/100 MHz clock rate).
6. Small size: the current implementation is sized 43 mm $\times$ 18 mm.
7. Low power: the current implementation consumes less than 200 mW at full operating speed (ASIC 78 mW). This value has to be seen in conjunction with the possibility to switch the system on and off very quickly.

The vision processing applied to the images is described in Sect. 2. The system architecture which enabled the FingerMouse to be built with the features mentioned above is described in Sect. 3. The resulting hardware system is shown in Sect. 4, and its performance compared to other systems in Sect. 5. Finally we present application scenarios for the FingerMouse in Sect. 6. An earlier version of this article was presented at the 19th international conference on architecture of computing systems (ARCS ’06) in Frankfurt am Main, Germany [3].

2 Image processing

2.1 Segmentation techniques

To achieve the hand/foreground segmentation, we evaluated several methods. Table 1 gives an overview

Table 1 Comparison between hand recognition algorithms

Method	Description, disadvantages
Static background subtraction	Two subsequently captured images are subtracted. Moving objects can be retrieved. *Disadvantages*: if the device is worn on the body, the whole scene is always in motion. The method is not applicable.
Color segmentation	The image is segmented by using a skin color tone reference. If a pixel is similar to the reference, it is classified to the hand [4]. *Disadvantages*: needs calibration to calculate the color reference. Changing lighting requires new calibration.
Active lighting	The foreground is illuminated by a light source of a specific spectrum. The image is captured through a spectral filter. Close objects should show a higher brightness [5]. *Disadvantages*: lighting requires a lot of power. Outdoors, the sunlight outperforms lighting over most of the spectrum.
Structured lighting	The scene is projected with a pattern, e.g. a dot grid. Captured images allow to determine the position of the reflected dots on the hand/foreground in 3D. Using a laser as a monochromatic light source, a narrowband capture filter, and a small lit area (the dots), we showed the feasibility of such a system against sun light with little power consumption [6]. *Disadvantages*: the resulting data is sparse, since limited to the number of projected dots.
Time of flight camera	A new type of camera sends out light, modulated over time. In the captured image, the phase of the modulation in the reflected light is determined, and leads to the distance of the captured object. *Disadvantages*: requires an active light source.
Contour tracking	A geometrical contour of the hand is tracked over time. In new images the tracker adjust the contourto follow the hand [7–9]. *Disadvantages*: needs calibration to initially put the contour on the hand. If the tracking is lost, new calibration is required.
Stereo vision image substraction	The image from two parallel cameras is subtracted. Since the background coincides, it disappears. This algorithm was implemented in older FingerMouse platforms (c.f. Sect. 5.1) [10]. *Disadvantages*: the foreground is badly retrieved as an overlay of two subtracted hands. Medium distanced objects produce a lot of noise.
Stereo vision depth mapping	Using two cameras, the depth of a scene can be computed. The hand is classified by its proximity to the cameras. This algorithm is integrated in the FingerMouse. *Disadvantages*: high computational effort.

of applicable vision methods and algorithms. It describes the disadvantages of each one, in the context of our wearable computing scenario. This scenario implies indoor and outdoor lighting conditions, a non-static camera position and a limited power consumption budget.

In the FingerMouse system, we use stereo vision depth mapping because of its good segmentation, outdoor applicability and the absence of calibration. It is combined with a color segmentation filtering step (optional), which is auto-calibrated by the stereo output. The problem of the high computation amount is addressed by a FingerMouse ASIC, specifically developed for the task (c.f. Sect. 3).

The computation techniques and mathematical models behind stereo vision depth mapping have been investigated and covered in literature since more than three decades (e.g. [11, 12]). While the principles of projective geometry on which the technique is founded have been described since the sixteenth century [13], real-time hardware implementations with TV-like resolution and frame rate became feasible and are described since the 1990s ([14–18], c.f. Sect. 5.2). The FingerMouse contains an implementation of a pixel based stereo correspondence analysis with block matching. The next subsection describes how it was implemented, adopted and extended to fulfill our wearable computing specific design goals.

2.2 FingerMouse vision algorithm brief

Two cameras with parallel viewing axes, offset by b (baseline), synchronously capture an image of the scene. For each pixel, a depth Z (also called range) can be computed, by comparing the two images. Objects in the two stereo image are horizontally translated by the so called disparity d (pixels) (c.f. Fig. 2). The disparity d of an object is inversely proportional to its distance Z from the cameras:

$$d = bf \frac{1}{Z} \frac{x_r}{x_s} \text{ (pixels)} \tag{1}$$

(f is the focal length of the lenses; x_s is the image sensor width [light sensitive area); x_r is the horizontal image resolution (pixels)].

The disparity is calculated for each pixel and then classified into foreground or background. To compute a disparity, a reference window, a block (3×5 pixels) around the pixel, is compared to blocks in the other stereo image, along a horizontal search window. The block with the highest correlation to the reference block gives the disparity (c.f. Fig. 2). Pixels with a disparity contained between the definable thresholds $d_{\text{near\_thresh}}$ (corresponding to $Z_{\min}$) and $d_{\text{far\_thresh}}$ (corresponding to $Z_{\max}$) are classified as hand/foreground, the other as background.

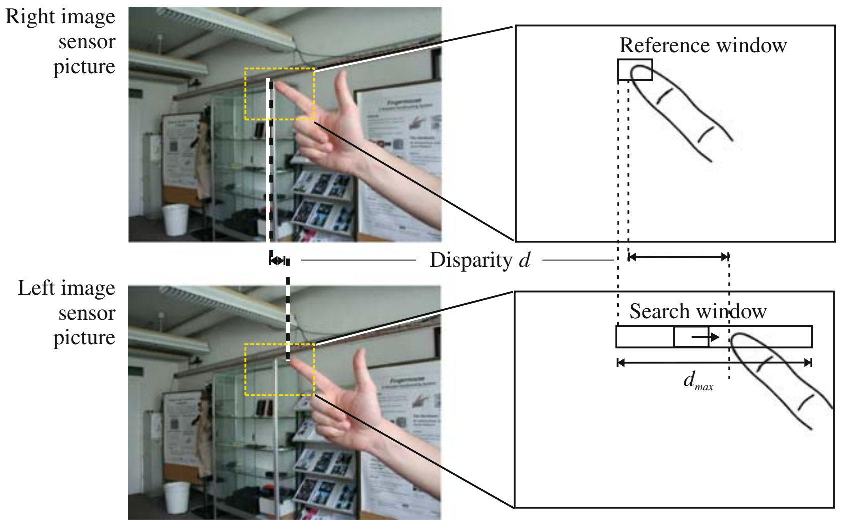


Fig. 2 Stereo input images and block matching scheme

The width of the horizontal search window is $d_{\max}$ and determines the highest disparity that can be measured. $d_{\max}$ also represents the amount of discrete values for the depth measurement and is often called disparity search depth. ($d_{\max} = 47$ pixels for the FingerMouse). The hand must not be closer than Z_{prox}, the distance corresponding to $d_{\max}$.

The correlation between blocks is calculated with two different similarity functions operating on the pixel intensity (brightness): the sum of absolute differences (SAD) and the census function. The census function maps a 3×3 block to eight bits, each bit indicating whether a border pixel is brighter than the center pixel. A 5×3 block is mapped to 3×8 bits, the combination of the results from the three 3×3 blocks inside the 5×3 block. Blocks are compared by the hamming distance of their census function. The SAD and census block comparisons have different noise characteristics when dealing with different kinds of area structure (from homogeneous to highly structured), and the census function is oblivious to absolute brightness.

The resulting measurements from a single correlation method yield noise. Figure 4 shows the distribution of measured disparities from a single matching run, and gives a hint on the noise level (e.g. all disparity measurements of $10 < d < 25$ are false, since no parts of the scene have the corresponding depth). This noise results from different sources, which are described in Sect. 2.3. It should be clear, that applying two distance thresholds reduces noise (noise 2 in Fig. 4), compared to only using $Z_{\min}$.

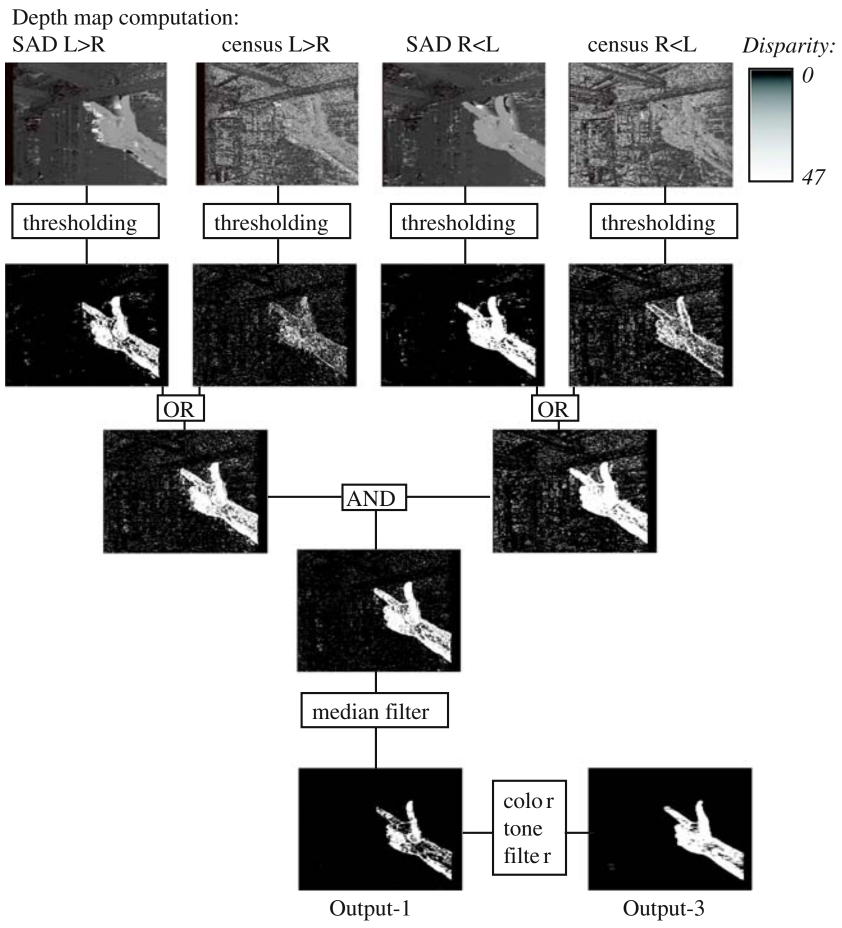
To deal with perspective occlusions, the correspondence analysis is performed both from left to right (L $\rightarrow$ R search) and from right to left (R $\rightarrow$ L search). Using the measured disparity information, the L $\rightarrow$ R search result is transformed to the right camera perspective, to be combined with the R $\rightarrow$ L results.

By combining the four redundant search runs (SAD and census, both R $\rightarrow$ L and L $\rightarrow$ R), noise is reduced. Remaining noise is filtered by a median filter. The resulting picture is output-1. The hand coordinates, output-2, are derived from a center-of-gravity computation. Figure 3 shows all the intermediary processed images and the final outputs.

Since the noise countermeasures focus on reducing noise in the background rather than noise in the hand/foreground, a color step helps to “fill up” the hand: the output-1 image is enhanced using color segmentation. A reference color tone (hue) is computed on-the-fly using the output-1 image (which shows parts of the hand and noise) and the buffered color tones. Pixels that have a similar color and are close to foreground pixels in output-1 are classified as foreground in the output-3 image. The corresponding hand coordinates are called output-4 (also see Fig. 9). Since the color segmentation is done on-the-fly, the color reference is always taken from the previous image. The use of the color fill-up step is optional.

Many other stereo vision algorithms perform a perspective rectification of the images, before running the correspondence analysis, to correct the effect of the two different camera perspectives onto the hand. Alternatively, the effect can be reduced by choosing a small camera baseline b, as in the FingerMouse. A rectification step was omitted since we prefer concurrent measures to consecutive ones. This reduces latency and is also necessary to free the system of a frame buffer (c.f. Sect. 3). Potentially, it could be added easily to the system.

Fig. 3 Image processing flow



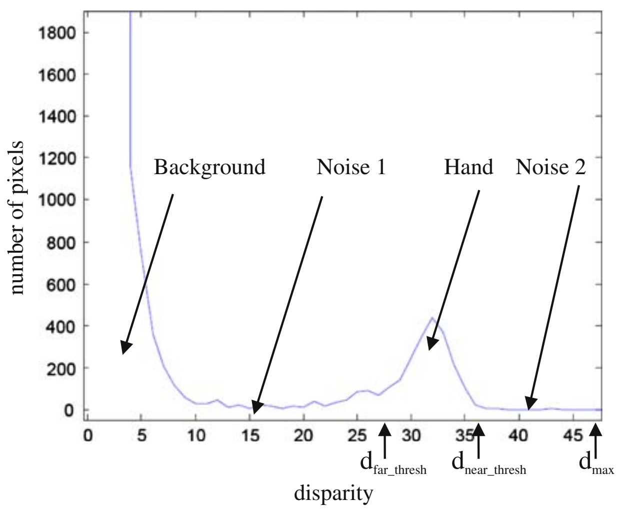


Fig. 4 Disparity distribution in a sample image containing hand and background

2.3 Robustness considerations

With the term robustness, we describe how the system's output quality is related to different challenges the FingerMouse faces in a wearable computing environment, a higher robustness meaning a better output quality for a specific scene.

A measure for output quality is to count the number of incorrectly classified pixels (noise). This noise can be divided into background noise (background pixels wrongfully detected as foreground) and foreground noise. To quantify the quality, the outputs are compared to ground truth values, e.g. a manually segmented bitmap.

When dealing with a specific application, the output quality of a post-processing stage (e.g. the tracking precision of the hand movement) is more relevant, and the quality measure has to be adopted. For example, in some applications, the correct detection of pixels along the contour of the foreground object is more important than of the pixels inside the object (which are harder to detect when the object has a uniform texture).

The factors that influence the FingerMouse's robustness all depend on the scene and its image acquisition. After the images have been digitally transmitted from the cameras, the rest of the system is deterministic, and can be simulated down to gate level.

The scene specific factors are:

- foreground objects occlude different background areas, respectively, in the two camera perspectives [19]
- close objects are projected in a different perspective onto the two image sensors
- homogenous areas lead to ambiguities, horizontally periodic structures confuse the block matching.

The image acquisition specific factors are:

- image capture can differ in the two cameras (e.g. global brightness)
- the image sensor limitations in extreme (low/high) lighting conditions
- the geometrical precision of the stereo camera setup.

In order to have the cameras produce images in an equal fashion, we synchronize their complete timing on clock cycle level. This reduces effects caused by different exposure moments, e.g. with artificial lighting, the scene brightness often slightly oscillates at 50/60 Hz. Remaining brightness differences are dealt with the bias independent census block comparison. In some scenes, the brightness of the light causes problems: if it is too high or too low, a limit in the sensor's dynamic range is reached, and structure is lost in under/overexposed image areas. In low lighting conditions, the sensors increase their signal amplification (gain) and/or apply higher exposure times, which leads to a decay in the signal/noise ratio and/or motion blur. When building the stereo setup, the cameras will never be exactly in parallel: while small translational errors have little effect, rotational errors in the three possible axes (horizontal pan, vertical pan, rotation in the image plane) affect the image processing. This rotational error can be minimized with a high precision manufacturing process or by later mechanical calibration, e.g. mounting the cameras on daughter boards with screws. Many image sensors also offer the possibility to transmit only a rectangular subset of the image (windowing). This permits to approximately correct small rotational errors in two axes (pan).

The system's robustness can be measured by simulating with different sets of input images and measure the resulting output quality. This also allows to evaluate candidate image sensors or to test different system parameters (e.g. camera baseline, focal length, etc.) with scenario specific test images before implementing the embedded device. The robustness against specific problems (e.g. angular geometry flaws, under/overexposure, noise, dynamic range) can be simulated by artificially modifying existing images of higher quality, or with artificially produced (e.g. by ray-tracing a 3D model) images.

Figures 5, 6 and 7 show the relative performance of the system under different scenarios, where the camera images have been artificially deteriorated (the curve legend in Fig. 5 is valid for all three figures). The input images are of VGA resolution, processed at half-VGA (320 × 480). The images are the same as in Fig. 2, containing a hand in the foreground. As a measure for output quality, we plotted the foreground recognition rate (♯foreground pixels recognized/♯foreground pixels in ground truth image) and background recognition rate for both the output with and without the optional color fill-up stage (output-3 and output-1). The tracking distance error is the distance in pixels between the measured center of gravity and the ground truth center of gravity. It is measured for both the filled and unfilled image (output-4 and output-2). The mean error distance for guessing the position is 277 pixels (in

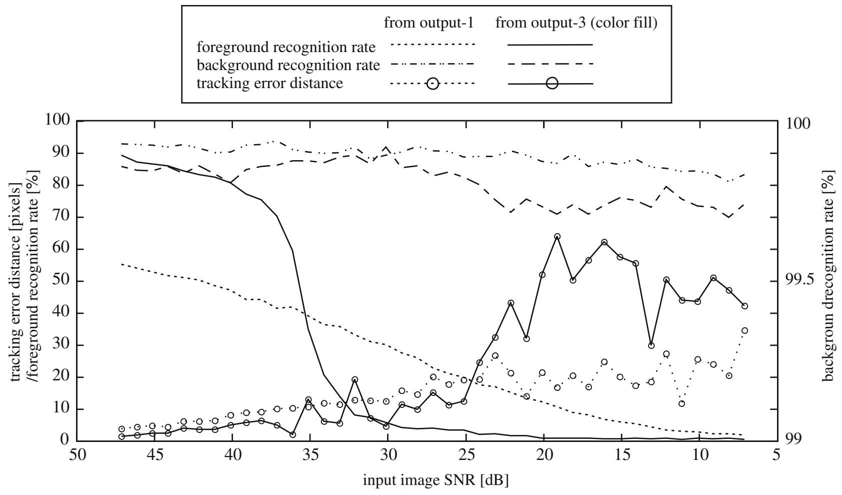


Fig. 5 Robustness against SNR decay (image sensor noise)

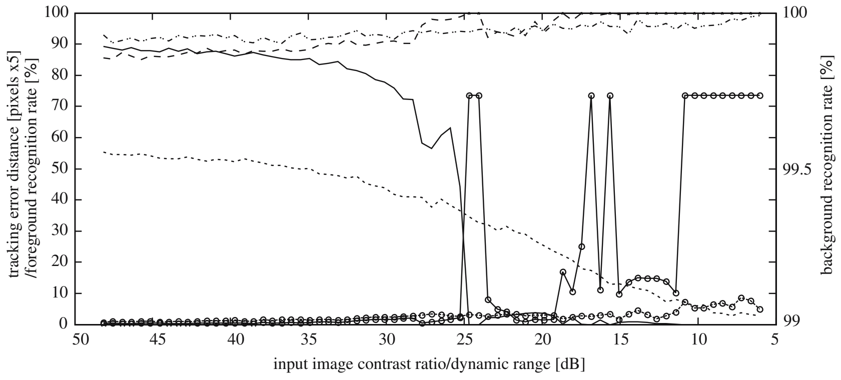


Fig. 6 Robustness against dynamic range decay (image constrast decrease)

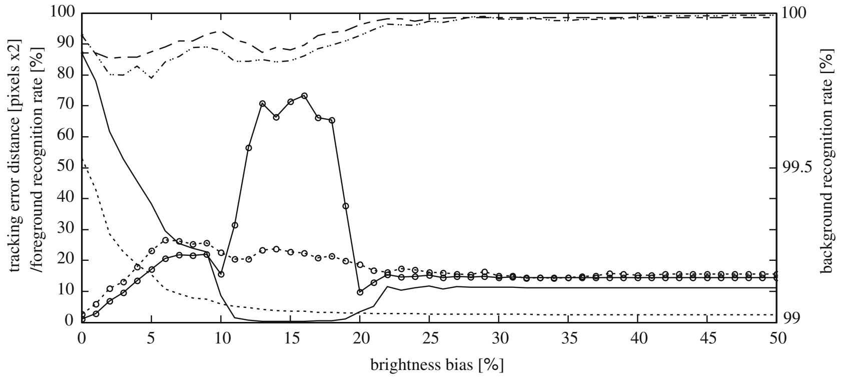


Fig. 7 Robustness against brightness bias (left image brighter)

half-VGA) (please note that its scaling is different in the three plots).

Figure 5 shows the system's reaction to more noise (lower SNR) in the sensor images. The artificial noise is uniformly distributed in both stereo images. In practice, noise occurs when lighting is low and the gain in the sensor is increased. The amount of noise also depends on the quality of the optics and the image sensor sensitivity. The input noise creates a linear decay in the (unfilled) hand recognition, which is caught up by the fill-up stage, until the latter breaks down after SNR < 35 dB.

Figure 6 shows how the system deals with a decreased dynamic range. The images were artificially decreased in contrast (over the whole image). This decay occurs when the image contrast is low or when the area of interest (e.g. hand, foreground object) has a low contrast (e.g. in the shade) within a high contrast scene (e.g. sun-lit). This time the unfilled foreground "disappears" much slower in the sample stereo set, and the filled output remains good until 28 dB of dynamic range.

Figure 7 shows what happens, when the left image is globally brighter than the right sensor image (a value of n% means, that an area of the same object is n% brighter). The system reacts very badly to the bias, but maintains some foreground recognition without generating background noise. In the synchronous camera system of the FingerMouse, the brightness bias is insignificant, but could occur when the used optics have different apertures, e.g. through manufacturing tolerances. Care has to be taken to avoid this.

The three figures show that the background stays mainly noise free (less than 0.5% noise pixels), even when the input images are flawed and when the hand area is not recognized any more. This makes it easier to determine the foreground area, even when the latter is noisy and is an effect of the filtering and depth map combinations. The foreground (output-1) is always partially captured, and allows to be used for post-processing, e.g. the center of gravity tracking, even under heavy constraints. Its filled version (output-3, and its position tracking, output-4) allows even a higher quality output (both segmented image and tracking), but its robustness is more limited compared to output-1 (and output-2) and falls behind after a certain amount of input image decay.

The actual input and output images of the presented figures can be found at http://www.fingermouse.ethz.c.

3 Hardware architecture

The architecture of the system and its components are shown in Fig. 8. The two cameras are arranged in parallel and they work synchronously with the ASIC and with each other. This is possible by clocking them through the FingerMouse-ASIC and by setting a synchronous operation start and identical timing through the microcontroller.

The images are not stored in a frame buffer, eliminating the need for an image-RAM. Instead, a window of only four (stereo-)lines from each stereo image is buffered inside the ASIC. The segmentation processing is done on-the-fly on the first three lines, while a new line is stored in the buffer. The segmented output is available concurrently with the image transmission from the cameras. The hand-tracking is also done on the fly, and the result is ready directly after the image transmission is done. This way, the result computation has a very small latency (the time between the transmission of a pixel and its computed output) of merely two row transmission times. The delay between a hand movement and the transmission of the hand pixels by the camera depends on the speed of the image sensor: the movement is integrated over the exposure time and then transmitted immediately (rolling shutter camera) or at a delay up to a full image transmission time, depending on its position in the image (global shutter camera).

To handle a pixel throughput of 5 M stereopixels per second and do online stereo matching, the ASIC has to perform over 20 G Operations per second [this value is based only the on stereo matching part, and only counting operations (subtract, accumulate, compare, fetch value) inside the loops, without any overhead. The complete figure on a RISC based implementation is much higher]. The disparity computation of one stereo pixel pair, which includes 188 block comparison operations ($d_{\max} \times 4$), is done within 16 ASIC clock cycles. This is possible through a high speed ring buffer, which stores the reference block and parts of the search window in a ring buffer, consisting of registers. During the 16 clock cycles, the ring buffer slides over the complete search window, fetching new pixels from the row-buffer RAM. At each clock cycle, 16 parallel units, directly wired to the ring buffer, do a block comparison. With some overhead and additional computations, this allows for a disparity search depth of $d_{\max} = 47$. The data and processing flow can be seen in Fig. 9.

The microcontroller controls the cameras, configures the ASIC and transports the tracking results via a RS232 interface. The segmented images (or optionally the SAD disparity maps) are directly output by the ASIC, over a 16-bit parallel/serial interface. The design could also be extended, so that the microcontroller could handle some or all post-processing of the images or could provide USB connectivity.

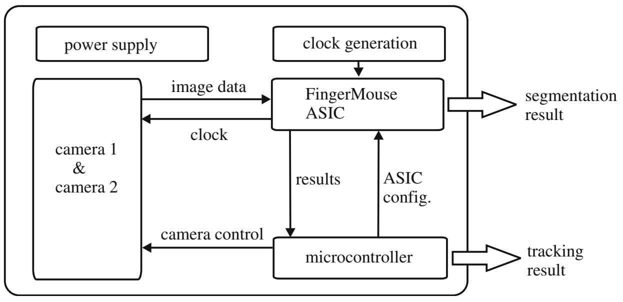


Fig. 8 System architecture

4 System implementation and results

The architecture is implemented on a four-layer PCB (size 43 mm × 18 mm). The FingerMouse ASIC die is directly bonded onto the PCB, without a package. Two image sensors (Omnivision OV7649, low voltage color CMOS VGA imager) in chip-scale package were used, together with small board-lenses. The system further

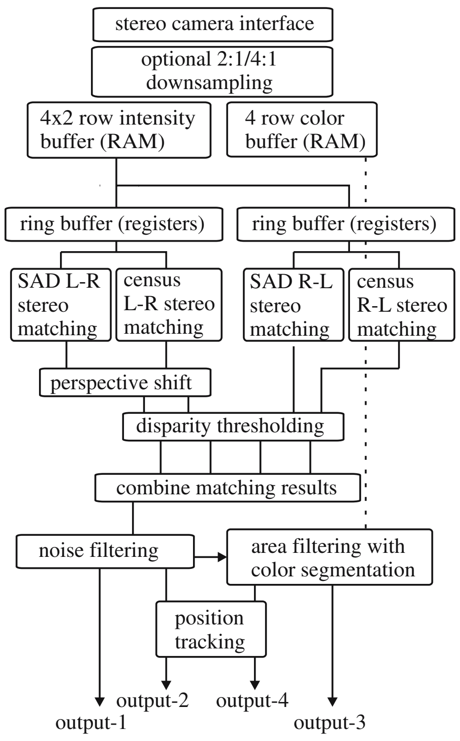


Fig. 9 Data path and processing inside the ASIC

includes an MSP430F1611 in a standard package (backside), and interfaces for communication. A battery voltage of 2.7–5.5 V is regulated to the different voltages needed onboard, using a single micro-power-management integrated circuit (MAX8620Y).

The (rolling shutter) cameras are arranged with with a baseline of 25 mm and deliver pictures of 640×480 resolution (processed as 320×480 in the FingerMouse) at a frame rate of 15 fps or 320×240 at 30 fps (at 80 MHz ASIC clock). The cameras can also output subset windows of the images. For example, when a horizontal position of an object in front of the FingerMouse needs to be determined, images of 320×3 pixels could be processed at several hundred fps. The delay between input and output pixels from the ASIC, running at 80 MHz, is smaller than 200 µs. The optics seen in Fig. 10 (left) are provided by Sekonix. With a focal length of $f = 3.6$ mm, the horizontal field of view of the FingerMouse has an opening angle of 53° (this corresponds to the viewing angle of a 35 mm film camera with a $f = 30$ mm lens). In this optical setup, the relation between the disparity d and the depth Z is as follows:

$$Z = 8\frac{1}{d}[m](Z_{\text{prox}} = 0.17\,\text{m}).$$

The ASIC was manufactured in May 2005, by UMC. Table 2 shows some figures of the FingerMouse ASIC. Table 3 gives an overview of the components' power consumptions. Figure 10(left) shows the FingerMouse PCB with the optics, no electronics mounted.

5 System performance comparison and discussion

5.1 Performance of the different FingerMouse implementations/architectures

To give a notion of the system's performance, we provide an overview over three other prototypes that have been implemented at our lab. They all use stereo vision to segment and track the hand, but the underlying processing architecture is different.

1. The first prototype system was built in 2001/2002. It uses a DSP and a very rudimentary algorithm: for the background/foreground segmentation, two corresponding stereo-vision pixels were compared (image subtraction) and classified via a threshold. In a second step, the resulting segmentation is filtered with morphological operations. This results in a ''dirty'' segmentation of the background and the overlay of the hands (translated in the stereo images) which was nevertheless usable for tracking. Image processing is done after complete image transmission, resulting in some latency (13.5 ms) [10].
2. The second prototype was developed in 2003/2004. It uses the same algorithm as the first prototype, with some further refinements. It uses an FPGA for the computation, which allowed zero latency and a much higher data rate than the first prototype, thanks to parallel processing and concurrent image transmission.
3. Implementation of the new algorithm on a desktop PC with additional optimizations. It uses two USB cameras and does not use the color segmentation processing step.

Even though the DSP/FPGA based FingerMouse prototypes run a much more rudimentary algorithm requiring less calculations per pixel, the efficiency comparison shows that the architecture based on our ASIC clearly outperforms the other architectures (Table 4).

5.2 Other real-time stereo systems

While stereo correspondence algorithms have been described earlier, real-time systems emerged in the 1990s. The following embedded hardware system have been implemented using FPGAs and DSPs:

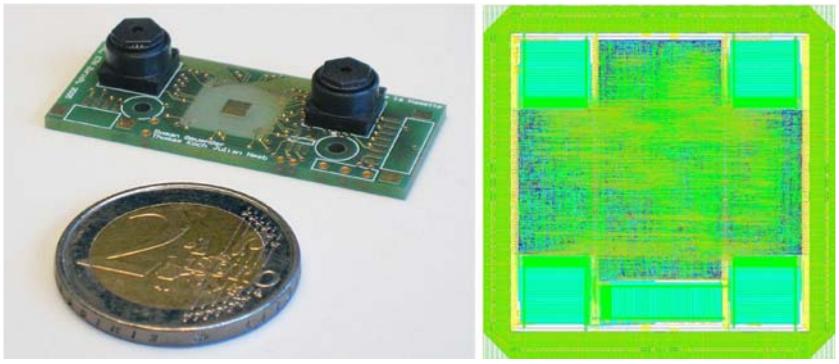


Fig. 10 FingerMouse prototype PCB (next to 2 Euro coin) (*left*). ASIC layout (*right*)

Table 2 FingerMouse ASIC specifications

Supply voltage	2.5/3.3 V (core/ I/O)
Chip size	2,227 μm × 2,227 μm (excluding sealring and bonding areas)
Chip technology	umcL250, 250 nm
Pin count	28 input; 22 output; 20 power; 14 empty
On-chip RAM	4 KB (3.432 KB used)
Transistor count (without RAM)	approx. 380,000
Pixel input format	B/W 8 bit/pixel or YUV 4:2:2 16 bit/pixel
Input image size and format	Any aspect ratio, any resolution up to the maximal image width: max. 1,360 (internal: max 340, after factor 1, 2 or 4 downsampling)
Image processing rate	5 Mpixel/s at 80 MHz /6 Mpixel/s at 100 MHz
Power dissipation at full processing rate	78 mW at 80 MHz and 96 mW at 100 MHz clock speed
Interfaces	Configuration, tracking results: RS232/segmented image output: 16-bit parallel/serial interface

Table 3 FingerMouse system power budget

Component	Power
FingerMouse ASIC	78 mW
Cameras	2 × 30 mW
MSP430	5 mW
Clock generator	23 mW
Total internal dissipation	166 mW
Power regulation effiency	89%
Total input power from battery	187 mW

1. The “stereo machine for video-rate dense depth mapping” ([17], 1996) performs 30 M block comparisons per second. (FingerMouse: 470–587 M) e.g. 200 × 200 images at 30 fps and a disparity range of 32. It features image rectification and can use 2–6 input cameras. The system uses an array of TMS320C40 DSP processors.
2. The “PARTS reconfigurable computer” ([15], 1997) is a general-purpose reconfigurable machine with wide I/O memory, consisting of 16 Xilinx 4025 FPGA’s and 16 one-megabyte SRAMs. It runs stereo vision at 2.3 GOp/s, resulting in a performance of 42 fps at 320 × 240 resolution and 24 disparity computations (15M block comparisons per second, 22 W power consumption).
3. An implementation on a hardware of several Xilinx 4013 FPGAs is described in [16] (1999). It achieves 30 256 × 256 frames per second, of a maximal disparity 32.

More embedded system can be found in [20–22].

Since a few years, real-time implementations are achieved on standard PCs. This has become possible through the increase of the PCs performance, the use of new processor technologies like MMX and SSE (on ×86 platforms), better cache and multi-core processors. The main performance increase results from algorithm optimizations though: since coherent objects in a a scene are coherently projected onto the image sensors, it is clear that the depth of a pixel is strongly correlated to its neighboring pixels. This is exploited in the correspondence analysis: the image is treated in different hierarchal steps, using a priori information from the previous step to decrease to amount of necessary block comparisons and thus increase speed. Some of those techniques are described in [23, 24].

Many different software implementations exist, e.g.:

1. Triclops SDK is a software by the company Point Grey Research (http://www.ptgrey.com). It does real-time stereo block matching in high quality on

Table 4 Comparison between the different systems

	DSP based FingerMouse	FPGA based FingerMouse	Desktop PC based implementation	ASIC based FingerMouse
Processing architecture	DSP (TI TMS320 VC33)	FPGA (Xlinix Spartan II)	PC (Intel Pentium 4)	FingerMouse-ASIC
Internal image res.	128 × 128	640 × 480	640 × 480	320 × 480
Clock	75 MHz	20 MHz	2.8 GHz	80 MHz
Power (only processing)	130 mW	1,000 mW	> 50 W	78 mW
PCB size	4,672 mm^2 73 mm × 64 mm	3,381 mm^2 69 mm × 49 mm	–	774 mm^2 43 mm × 18 mm
Thickness PCB and optics	35 mm	45 mm	–	8 mm
Segmentation quality	Low	Low	High	High
Output latency (ms)	13.5	< 1	1,000	< 1
Image data rate	0.5 Mpixels/s	10 Mpixels/s	0.6 Mpixels/s	5 Mpixels/s
Efficiency (E/pixel) (μJ)	0.26	0.1	> 83	0.016

a PC and can be used with Bumblebee, a stereo camera head by the same company. These pre-calibrated products allow fast prototyping of depth computing applications. According to the manufacturer's website (1 May 2006), the software achieves a frame rate of 31 fps, at 320 × 240 resolution and a disparity depth of 48 (similar than the FingerMouse running at 80 MHz), running on a Pentium IV, 2.4 GHz.

2. E-stereo is a "C++ library for real-time disparity estimation. The library contains various functions for dense stereo matching from 2 to 3 rectified images and 3D scene reconstruction", by David Demirdjian [25]. On a Pentium IV, 1.7 GHz, with 320 × 240 input images, a disparity depth of 32, it processes 14–18 fps, according to the author.
3. GPU based stereo vision. Some projects use the processing power of a PCs graphical processor to run stereo block matching algorithms, e.g. in [26] (2003), a NVIDIA GeForce4 graphics card in a PC achieves 50–70 M block comparisons per second (FingerMouse 470–587 M).

5.3 Discussion

The six described systems perform similar stereo block matching than the FingerMouse, and could also be used for foreground segmentation. Since their original task is to deliver depth maps, the systems have been optimized for depth output quality, while the refinements in the FingerMouse are more focused on the segmentation of a hand.

The software based PC solutions offer a higher level of flexibility in system design and implementation. The image throughput performance of the PC solutions is equalling the FingerMouse, when using today's state-of-the-art desktop or notebook processors.

Their disadvantages are:

- Higher latency: the images have to be transmitted from the cameras first, passing through an interface chip, a driver software and a video API. The full image has to be stored in RAM before an optimized algorithm can process it. Due to the algorithmic complexity, the latency is not deterministic any more. While latencies over 100 ms are tolerable for some applications, they decrease usability in human–computer interaction with real-time visual interaction (like moving a mouse pointer on a screen).
- Little mobility: a notebook solution is somehow portable, but truly mobile platforms like wearable computers (e.g. the QBIC [27]) remain a few years behind when it comes to processing speed, due to their lower power budgets.
- Less power efficiency: the general purpose architecture of the computers results in a lower processing efficiency (throughput/energy consumption). Additionally, the vision processing has to share the cpu performance with other applications running concurrently.

The other embedded implementations also achieve low latencies, but the FingerMouse has the advantage of its tightly coupled parallel computation and memory architecture inside the ASIC, and thus performs better.

Although the described systems are far from identical, the numbers should give an impression of the performance of the FingerMouse compared to other systems:

The FingerMouse outperforms its competitors on the performance/power ratio, the latency, total system power consumption and size.
The FingerMouse falls back when it comes to flexibility and depth map quality (depth maps are not used in our application scenarios).

6 Application scenarios

In most vision processing that analyzes an object in a scene, the segmentation of the object in the image is a crucial step. It is a standard task and is trivial when the camera position is static. When the system is in motion, and power consumption, size or latency are crucial, a device like the FingerMouse is useful. The following scenario examples describe what the mobile segmentation could serve for.

Section 6.1 shows the driving scenario and nomenclature origin of the FingerMouse. Sections 6.2 and 6.3 show further theoretical application scenarios for the FingerMouse smart camera system, including actually processed images.

6.1 Wearable computing HCI

Worn on the body, the FingerMouse allows the user to control a wearable computer with his hands.

The segmented output can be used to recognize gestures (c.f. [28]) or extract other information (e.g. orientation of the hand, position of the fingertip, silhouette generation, etc.). A basic position tracking of the foreground area is already performed inside the FingerMouse.

Thanks to its size, the FingerMouse could also be attached to the head-up display, in order to allow the user to point directly into the virtual screen in front of him. A drawback is, that current head-up displays have a smaller field-of view, that would only cover a subset of the FingerMouse field of view, and thus reduce the usable pointing resolution. A countermeasure is to increase the focal length of the optics in the FingerMouse, but this also increases the minimum distance of the hand Z_{prox}, unless the baseline b is also reduced (which cannot be changed for a given system, in contrast to the lenses) (Fig. 11).

6.2 Mobile video telephony with background removal

In mobile video telephony, the FingerMouse could replace the camera, and offer the possibility to remove or blur the background in an image. To cover the user in the image, a silhouette around the detected foreground (output-1) is computed and slightly enlarged, and then combined with the original right camera image.

The background removal improves user's privacy (hiding his current environment and location) and the privacy of other persons, since otherwise those could be filmed in the background during the video communication (c.f. Fig. 12). Furthermore, the bit rate of the video stream is reduced (or its quality improved for a given bit rate).

6.3 Visual augmentation and hazard detection

The FingerMouse could help provide simple vision augmentation. Figure 13 shows, how a FingerMouse mounted on a shoe segments objects in front of the user. Triggered by the foot stepping down, the system could give an audio cue to warn the user, if any object lies in his trajectory or even hint to the relative position (left/right/straight ahead) and distance of the object to the shoe. The object distance can be measured by averaging (for sub-pixel disparity precision) the disparity values of the detected foreground. Since the foot is not moving for a short period of time, several images can be taken, e.g. with different distance thresholds, to gain more information about the object. The necessary post-processing of the 1-bit output images (only 9,600 bytes, in QVGA) could be handled by a microcontroller (or by a wearable computer). Such a system and its battery can be fitted into a shoe and could operate non-stop for a whole day.

The system in the shoe could provide walking hazard detection to visually impaired people or to wearable computing users, walking while focusing on a head-up display. In a similar fashion, the FingerMouse could help a robot avoid or follow objects.

7 Conclusion and outlook

We proposed an architecture for a fully integrated real-time vision capturing and processing system that achieves a high performance although working under strict constraints. The small size and low power consumption of the implementation qualifies for use within a wearable system. With its figures of size, power consumption, performance and latency, the FingerMouse system is unrivaled today.

7.1 Outlook

The size of such a system could certainly be further reduced, when resorting to more complex construction techniques, like multi-chip modules. On the other hand, the size reduction is limited by the need of an offset between the two stereo cameras. Reducing this offset would influence the vision processing, altering the depth measuring range.

A further reduction in power consumption is still possible. Switching from 250 to sub 100 nm CMOS technology allows for a decrease of the ASIC's power

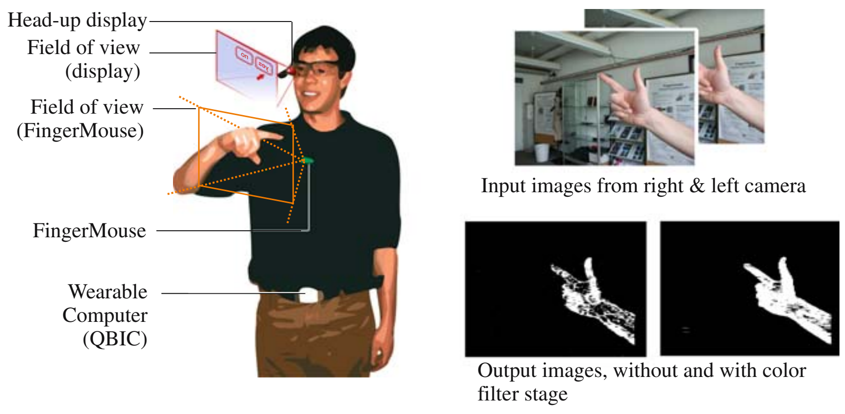

Head-up display
Field of view (display)
Field of view (FingerMouse)
FingerMouse
Wearable Computer (QBIC)
Input images from right & left camera
Output images, without and with color filter stage

Fig. 11 The FingerMouse as an input device to wearable computers

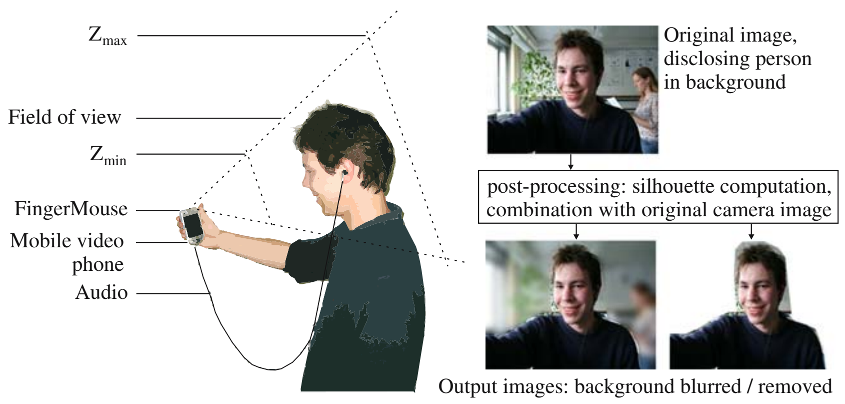

Z_{max}
Field of view
Z_{min}
FingerMouse
Mobile video phone
Audio
Original image, disclosing person in background
post-processing: silhouette computation, combination with original camera image
Output images: background blurred / removed

Fig. 12 Mobile video telephony scenario

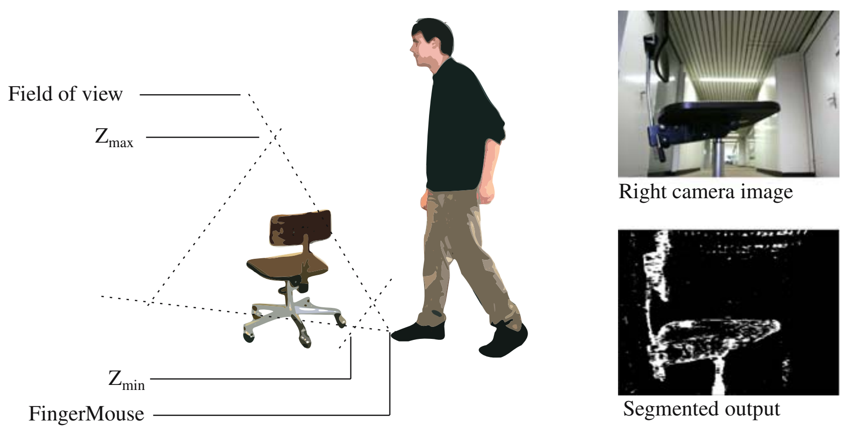


Fig. 13 Hazard detection scenario

dissipation by a factor of five without degradation of the computing performance. The trend in CMOS camera development also shows dropping power consumption while sensor performance is increasing.

The current size indicates that autonomous vision processing devices could already be used in wearable systems, and could even be integrated into mobile phones or PDA's, as described in the application scenarios.

Ongoing information and test images can be found at the FingerMouse project homepage (http://www.fingermouse.ethz.ch).

Acknowledgments We would like to thank Andreas Burg and Marc Wegmüller for their assistance in the ASIC design process. Most of the hardware implementations of the new prototype were done within student projects. Therefore, we thank the students Roman Gmünder, Julian Heeb, Thomas Koch and Sven Kuonen for their hard work. We also thank Martin Lanz, for his assistance with the die-bonding. For her help in the scenario photography, we thank Claire Muller.

References

1. Crowley JL, Coutaz J, Bérard F (2000) Perceptual user interfaces: things that see. Commun ACM 43:54–ff
2. Card SK, Newell A, Moran TP (1983) The psychology of human–computer interaction. Lawrence Erlbaum, Mahwah
3. de la Hamette P, Tröster G (2006) Fingermouse—a button size visual hand tracking and segmentation device. In: Proceedings of the 19th international conference on architecture of computing systems-ARCS 2006, pp 31–41
4. Habili N, Lim CC, Moini A (2004) Segmentation of the face and hands in sign language video sequences using color and motion cues. IEEE Trans Circ Syst Video Technol
5. Gandy M, Starner T, Auxier J, Ashbrook D (2000) The gesture pendant: a self-illuminating, wearable, infrared computer vision system for home automation control and medical monitoring. In: ISWC '00—Proceedings of the 4th IEEE international symposium on wearable computers. IEEE Computer Society, Washington DC, pp 87
6. de la Hamette P, von Waldkirch M, Tröster G (2004) Laser triangulation as a means of robust visual input for wearable computers. In: ISWC '04—Proceedings of the 4th IEEE international symposium on wearable computers, Doctoral Colloquium. IEEE Computer Society, Washington DC, pp 18–20
7. Blake A, Isard M (1998) Active contours, 1st edn. Springer Berlin Heidelberg, New York, ISBN: 3-540-76217-5
8. Grewal, MS, Andrews AS (1993) Kalman filtering: theory and practice. Prentice-Hall, Englewood Cliffs
9. Blake A, Curwen R, Andrew Z (1993) A framework for spatio-temporal control in the tracking of visual contours. Int J Comput Vis
10. de la Hamette P, Lukowicz P, Tröster G, Svoboda T (2002) Fingermouse: awearable hand tracking system. In: Adjunct

proceedings of the 4th international conference on ubiquitous computing

11. Hartley RI, Zisserman A (2004) Multiple view geometry in computer vision, 2nd edn. Cambridge University Press, London, ISBN: 0521540518
12. Faugeras O (1999) Three-dimensional computer vision: a geometric viewpoint, 3rd edn. MIT Press, Cambridge. ISBN: 0-262-06158-9
13. Frisius G (1544) Camera obscura. Historic publication on projective camera geometry
14. Porter RB, Bergmann NW (1997) A generic implementation framework for fpga based stereo matching. IEEE TENCON
15. Woodfill J, Herzen BV (1997) Real-time stereo vision on the parts reconfigurable computer. The 5th annual IEEE symposium on FPGAs for custom computing machines
16. Corke P, Dunn P (1999) Frame-rate stereopsis using non-parametric transforms and programmable logic. In: IEEE international conference on robotics and automation
17. Kanade T, Yoshida A, Oda K, Kano H, Tanaka M (1996) A stereo machine for video-rate dense depth mapping and its new applications. In: IEEE computer society conference on computer vision and pattern recognition
18. Konolige K (1997) Small vision systems: hardware and implementation. Eighth international symposium on robotics
19. Egnal G, Wildes RP (2002) Detecting binocular half-occlusions: empirical comparisons of five approaches. IEEE Trans Pattern Anal Mach Intell
20. Ayache N, Lustman F (1991) Trinocular stereo vision for robotics. IEEE Trans Pattern Anal Mach Intell
21. Fua P (1991) A parallel stereo algorithm that produces dense depth maps and preserves image features. Technical report 1369, Unite de Recherche, INRIA-Sophia Antipolis
22. Faugeras O, Hotz B, Mathieu H et al (1993) Real time correlation-based stereo: algorithm, implementations and applications. Technical report 2013, INRIA
23. Anandan P (1989) A computational framework and an algorithm for the measurement of visual motion. Int J Comput Vis 283–310
24. Ohm JR, Grüneberg K, Izquierdo E, Karl M Real time hardware system for stereoscopic videoconference with viewpoint adaptation. Image Commun 14, special issue on 3D technology
25. Demirdjian D Estereo: a c++ library for real-time stereo estimation. (http://www.sourceforge.net/projects/estereo/)
26. Yang R, Pollefeys M (2003) Multi-resolution real-time stereo on commodity graphics hardware. In: Proceedings of the IEEE computer society conference on computer vision and pattern recognition, 2003
27. Amft O, Lauffer M, Ossevoort S, Macaluso F, Lukowicz P, Tröster G (2004) Design of the qbic wearable computing platform. In: ASAP 2004—Proceedings of the 15th IEEE international conference on application-specific systems, architectures and processors
28. Starner T, Weaver J, Pentland A (1997) A wearable computer based american sign language recognizer. In: Digest of papers, first international symposium on wearable computers