Embedded stereo vision systems for mobile human-computer interaction

Author(s):
Hamette, Patrick de la

Publication Date:
2008

Permanent Link:
https://doi.org/10.3929/ethz-a-005787259

Rights / License:
In Copyright - Non-Commercial Use Permitted
Embedded Stereo Vision Systems for Mobile Human-Computer Interaction

A dissertation submitted to

ETH ZÜRICH

for the degree of
DOCTOR OF SCIENCES

presented by

PATRICK DE LA HAMETTE

Dipl. El.-Ing. ETH
born 7 November 1976
citizen of the Grand Duchy of Luxembourg

accepted on the recommendation of

Prof. Dr. Gerhard Tröster, examiner
Prof. Dr. Paul Lukowicz, co-examiner

2008
Acknowledgments

First of all, I would like to thank my academic supervisor Prof. Dr. Gerhard Tröster for giving me the opportunity to conduct research within the creative environment of the wearable computing research group. His enduring support made the success of the FingerMouse project possible.

Many thanks also go to my co-advisor Prof. Dr. Paul Lukowicz. His ideas and visions contributed to the creation of the FingerMouse project.

Since its beginning, the project was supported by many creative minds at ETH Zurich and I would like to thank Pascal Flammant, my long time friend with whom I realized the first prototype. Over the years, the knowledge and cooperation of fellow researchers went into the project and I give my thanks to Tomas Svoboda, Matthias Dyer, Andreas Griesser, Benjamin Knörlein, Andreas Burg, Felix Bürgin and Marc Wegmüller. Special thanks go to Prof. Dr. Cary Kornfeld, a friend and source of inspiration.

I also owe thanks to all the students that worked with me on the project and contributed to its results, namely Marco Lüthi, Claudio Semini, Dani Roth, Simon Jäger, Simon Haegler, Simon Schneiter, Leonardo Leone, Michel Krebs, Thierry Gschwind, Roman Gmünder, Julian Heeb, Thomas Koch and Sven Kuonen.

During my stay the ETH Electronics Laboratory, my many colleagues made the environment a pleasant one. I especially thank Ruth Zähringer, Clemens Lombriser, Daniel Roggen and Andreas Bulling for their support during difficult times.

Special thanks go to my friends Claire, Laurent, Pascal, Roland, Nicole, Petra, Gaston, Josiane, and many more, who gave me strength to fulfill my tasks and who made my life in Zurich great.

Finally, I thank my parents Charlotte and Armand for always supporting me in every aspect and for making it possible to fulfill my dreams in life.

Zurich, November 2008

Patrick de la Hamette
Abstract

Wearable Computers are a new generation of computers. They are worn on the body of the user, or are integrated in his clothes. Wearable Computers function as smart digital assistants, supporting the user at work or in every day tasks. What differs most from other mobile computers, such as smartphones, is the interaction between the user and the computer. This human-computer interaction (HCI) ideally does not constrain the user in his other tasks. The wearable computer either knows what the user needs by evaluating data from sensors in his textiles or a direct interaction takes place. Such direct interaction is needed, e.g., when the user interacts with a graphical user interface, as is delivered by wearable head-mounted displays. In that case, a pointing device is needed.

In this thesis, we present such an input device: the FingerMouse. It is a stereo camera system with integrated real time processing of images. When worn on the body, the system captures the user’s hand, and is able to extract the position of the hand, as well as a 1-bit image of the hand without the background (foreground segmentation).

The system processes images using a specifically designed integrated circuit (IC). It runs a foreground classification algorithm that is derived from area based stereo vision matching, a class of algorithms able to extract ranges of pixels in a scene. Such algorithms are applied since more than 30 years, but require a great effort of computing if running in real time. The algorithm used in the FingerMouse has been adopted for the purpose of the 1-bit classification of foreground and background, under the assumption that the foreground object is the closest object to the camera. Additionally, we combine the output with color based classifications. The proposed algorithm does not need any initialization.

This algorithm runs as an IC consuming only 78 mW, while processing 5 million stereo pixels per second, e.g. 60 frames per second of 320 × 240 pixels resolution. The complete system consumes 187 mW, and is realized in a small form factor: 43 mm × 18 mm. Nor the system size, nor the total power consumption and nor the pixel throughput per power are achieved by other available stereo systems in 2008.

The main contributions shown in this thesis are the design of the adopted stereo vision algorithm, an evaluation of this algorithm in a visual environment according to the FingerMouse system’s applications, and finally the implementation of our approach into a state-of-the-art embedded stereo camera system.
Zusammenfassung


in der Kameraaufnahme verdecken. Das Verfahren kombiniert ausser-
dem eine zusätzliche Klassifizierung, welche Pixel anhand der Farbe
von Objekten im Vordergrund (z.B. Hand) segmentiert. Das Verfahren
erlaubt zudem einen Systemstart ohne Initialisierungphase.

Das entwickelte Verfahren läuft in Form eines integrierten Schaltkreises
(IC) welcher nur 78 mW Leistung verbraucht, gleichzeitig aber Pixel-
raten von 5 Millionen Pixeln pro Sekunde verarbeitet. So können z.B.
60 Bilder pro Sekunde bei einer Auflösung von 320 × 240 Pixeln ve-
rarbeitet werden. Das gesamte FingerMouse System verbraucht 187
mW und wurde in miniaturisierter Form gebaut, es misst 43 mm × 18
mm. Sowohl die niedrige Leistungsaufnahme, als auch die Pixelrate im
Verhältnis zur Leistung, und die Miniaturisierung übertreiben andere
existierende Stereokamerasysteme in 2008.

Die wichtigsten Beiträge welche in dieser Arbeit präsentiert werden
sind der Aufbau des angepassten Stereo Bildanalyse Verfahrens, eine
Evaluation dieses Verfahrens anhand von Simulationen mit Testbildern
aus HCI-Szenarien und die Umsetzung des Verfahrens in ein state-of-
the-art Stereo Kamera System in Form eines eingebetteten elektronis-
chen Systems.
# Table of Contents

1. **Introduction**  
   1.1. Wearable Computing ................................. 2  
   1.2. Human Computer Interaction in Wearable Computing and Mobile Devices ................................. 3  
   1.3. The FingerMouse Project ........................... 5  
      1.3.1. Research Objectives ........................... 5  
      1.3.2. Approach ................................... 5  
   1.4. Outline of the Thesis ............................... 6  

2. **FingerMouse System Description** .................. 9  
   2.1. System Description ................................ 10  
      2.1.1. Functionality of the System .................. 10  
      2.1.2. The FingerMouse Prototypes .................. 13  
   2.2. Application Scenarios .............................. 14  
      2.2.1. Wearable Computing HCI ....................... 14  
      2.2.2. Mobile Video Telephony with Background Removal ....................... 15  

3. **Algorithm for Background/Foreground Segmentation** 19  
   3.1. Introduction: Algorithm Overview .................. 20  
   3.2. Depth Mapping Step ................................ 22  
      3.2.1. Specification of the Implemented Area-Based Matching .................. 23  
      3.2.2. Characterization of False Depth Measurements .............. 30  
   3.3. Foreground Segmentation ............................ 32  
   3.4. Fusion to a Single Segmentation Map ............... 36  
      3.4.1. Left-Right Perspective Conversion of Disparity Measurements .................. 36  
      3.4.2. Fusion ..................................... 37  
   3.5. Noise Filtering .................................... 41  
   3.6. Fusion with Color Segmentation Module ............. 45  
   3.7. Foreground Position Measurement ................... 51  
   3.8. Conclusions ....................................... 52
## 4. Evaluation in an Uncontrolled Environment

4.1. System Output Quality Quantification ............................................. 56
   4.1.1. Segmentation Quality ................................................. 56
   4.1.2. Tracking Precision ................................................ 57
   4.1.3. Robustness ............................................................ 57
4.2. System Evaluation for Application Scenarios .................................. 57
   4.2.1. Motivation ............................................................. 57
   4.2.2. Simulation Setup .................................................... 57
   4.2.3. Simulation Results ................................................ 59
4.3. Influence of Baseline and Focal Length ...................................... 64
   4.3.1. Motivation ............................................................. 64
   4.3.2. Simulation Setup .................................................... 64
   4.3.3. Simulation Results ................................................ 65
4.4. Impact of Camera Alignment Tolerances ...................................... 69
   4.4.1. Motivation ............................................................. 69
   4.4.2. Simulation Setup .................................................... 69
   4.4.3. Simulation Results ................................................ 70
4.5. Sensitivity to Different Lighting Situations .................................. 73
   4.5.1. Motivation ............................................................. 73
   4.5.2. Simulation Setup .................................................... 74
   4.5.3. Simulation Results ................................................ 75
4.6. Sensitivity to Radiometric Bias ................................................ 80
   4.6.1. Motivation ............................................................. 80
   4.6.2. Simulation Setup .................................................... 80
   4.6.3. Simulation Results ................................................ 80

## 5. Embedded Implementation

5.1. Architecture of the FingerMouse-IC ............................................. 84
   5.1.1. Overview of IC architecture ........................................ 84
   5.1.2. Image Input Layer ................................................... 84
   5.1.3. Depth Mapping Layer ............................................... 88
   5.1.4. Segmentation Processing Layer .................................... 91
   5.1.5. Interfaces .............................................................. 95
   5.1.6. FingerMouse-IC Prototype Specifications ....................... 95
5.2. Architecture of the Embedded FingerMouse System .......................... 97
   5.2.1. Optical Setup ....................................................... 97
   5.2.2. Embedded FingerMouse System ................................... 101
5.3. Other Real-Time Stereo Systems and Comparison ........................... 104
   5.3.1. Earlier FingerMouse Prototypes (DSP, FPGA based) ............. 104
1 Introduction

This introduction starts with a glance at the field of wearable computing and at human computer interaction (HCI) research related to wearable and mobile computing. The FingerMouse project, basis of the thesis, is introduced. The research objectives and approach are laid out.
1.1. Wearable Computing

As a new generation of computers, wearable computers are worn on the user’s body or are integrated in his textiles (Kirstein et al., 2002, [1]; Tröster et al., 2003, [2]). This enables new application scenarios: the computer becomes a mobile digital assistant helping the user perform certain tasks. In contrast to other mobile devices (laptops, PDAs, smart phones), wearable computing systems allow applications where the user is fully mobile and keeps his hands free (cf. Mann et al., 2001, [3]; Weiser et al., 1999, [4]). Applications that need user input rely on different sorts of user interaction. Ideally, the system inherently derives user input by context sensing (cf. Clarkson et al., 2000, [5]; Abowd et al., 1998, [6]). In other applications, explicit user input is required, and is performed using different wearable devices. Some of those devices occupy the user’s hand (e.g. mobile keyboards), others leave the user’s hands free, e.g. voice recognition systems.

An example of a wearable computer is the ETH QBIC, a belt-integrated computer system, shown in figure 1.1 ([7]). The QBIC system shares the same technical capabilities as today’s laptop computers and smart phones: mobility, battery operation and wireless communication.

![Figure 1.1: The ETH QBIC wearable computer](image-url)
1.2. Human Computer Interaction in Wearable Computing and Mobile Devices

Human Computer Interaction\(^1\) (HCI) in wearable systems offers a wide range of new user input modalities.

Context sensing is a methodology of retrieving input from the human (the user), without requiring explicit user interaction. Sensors distributed in the textiles retrieve information that is analyzed to understand the user’s actions and react accordingly.

For explicit user input, a range of new devices are available for wearable computers. Ideally, even the explicit user input does not obstruct the user more than necessary.

The tracking of the user’s hand is a common modality for user input, and the mouse is a standard input device for desktop computers. In wearable applications, it is comfortable if the bare hand of the user can be tracked, without the need of taking a device into the hand. A tracking of hand movements in three dimensions can be achieved using acceleration sensors (e.g. the Lightglove system by Howard et al., 2001, [9]) or ultrasonic tracking (cf. Stiefmeier et al., 2006, [10]). Another popular approach is the use of computer vision. It promises not only to track the hand movement, but also to capture images of actual hand shapes.

A fundamental problem in machine vision based hand tracking is the distinction between the hand area in the image and the rest of the scene. This problem can be solved easily for static cameras, since the background is known. It is hard to do if the camera is in motion, as is the case in wearable devices.

Numerous machine vision approaches exist, to classify the hand area in a captured image of a scene:

- color segmentation

\(^1\)“Human-computer interaction is a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them.” (Definition by Raskin et al., 1994, [8])
• active lighting
  Active lighting systems illuminate the scene, e.g. with IR light. The scene is then captured through an IR filter. The hand, being closest to the camera, reflects most light and is thus distinguished by its brightness. Such systems cannot be used outdoors as the sun produces too much IR light itself.
  The ”Gesture Pendant” is an implementation of such a system (Gandy, Starner et al., 2000, [16]). Another such system was presented by Lee et al. (2002, [17]).

• structured lighting
  This approach relies on light patterns projected onto the hand. These patterns can be retrieved from the camera images, and each part of a pattern mapped to a 3D position.
  Many research implementations of such systems exist. Balch et al. present a system robust against strong sun illumination (1997, [18]), as well as Bang et al.(2003, [19]). A system was also developed within the FingerMouse project (de la Hamette et al., 2004, [20]).

• time of flight cameras
  T.o.f-cameras are able to capture ranges of pixels, using a single lens. They emit light and measure the time the light takes to be reflected. The t.o.f. cameras work indoors only. Cf. Lange et al.(2003, [21]), Büttgen et al.(2001, [22]).

• contour tracking
  This method relies on the contrast at the transition between the hand and the background. A set of the transitions model the contour of the hand and are tracked over time. Cf. Blake et al.(1998, [23]); 1993, [24]).

• stereo vision
  This approach is the base of the algorithm used in this thesis. The stereo vision algorithm computes correspondences between pixels in two images, captured by two parallel cameras. Range data can then be computed for each pixel, but the method requires great computational effort.
1.3. The FingerMouse Project

In the field of wearable computing and vision-based HCI, much research work is focussed on the extraction of higher level information, such as hand gestures. When it comes to the low-level tasks, such as segmentation against the background, many projects rely on standard vision approaches such as color segmentation, implemented on standard computer platforms, such as the hand tracking and gesture recognition projects for wearable computers by Liu et al. (2004, [25]) and Westeyn et al. (2003, [26]).

The FingerMouse project aims to develop a HCI system in embedded electronics, integrating all components from image capture to measurement output, based on the stereo vision method. The project approaches the machine vision task bottom-up, with most focus on the low-level stages of the image processing pipeline, such as image acquisition and foreground detection. The goal is an embedded stereo vision system that solves the task within the requirements of wearable computing: self-contained, battery powered and small, while operable in a multitude of uncontrolled environments.

The main achievement, a low-power solution suitable for wearable applications, produces (among other data) segmentation maps that can be used for all mobile vision applications relying on foreground segmentation (e.g. wearable color trackers or IR systems such as the GesturePendant by Gandy et al., 2000, [16]).

1.3.1. Research Objectives

The following aims are being pursued in this Ph.D thesis:

- Evaluation of different vision methodologies for tracking a hand and detecting shapes.
- Development and experimental evaluation of hardware prototypes of integrated, autonomous wearable vision devices.
- Investigation on methods of making the systems robust, and to quantify their robustness.

1.3.2. Approach

Within the vision process from the image capture to the final results, one step deals with the classification of a scene into foreground and
background objects, the so called segmentation. Once the user’s hand is segmented from the rest of the captured scene, features, such as the motion of the hand and orientation can be retrieved in a straight forward fashion. Other interaction modalities, like hand gesture recognition, require further high level algorithms, which rely on foreground segmentation to extract the higher level features.

This thesis aims to build and investigate upon integrated hardware vision modules that perform the segmentation task in an autonomous way. Those systems are able to perform two dimensional position tracking but also can provide a segmented image for further processing in a mobile device (e.g. a wearable computer). While foreground segmentation is a standard task for immobile camera systems and a workstation running the algorithm, it becomes a challenge in the wearable environment, due to the indoor and outdoor environments and the constraints in size and in power consumption.

The FingerMouse project can be divided into several steps:

- investigation on foreground segmentation algorithms and evaluation of these algorithms in a software test environment using prerecorded images.

- implementation of the foreground segmentation algorithms in embedded electronic modules, based on DSPs, FPGAs and also in an integrated circuit. A focus lies on power consumption, small size and low latency of the measurements.

- evaluation of methodologies to make the vision process more robust for applications in a mobile environment. Approaches such as stereo vision, active vision and the use of special image sensors and additional sensors are under consideration.

1.4. Outline of the Thesis

Chapter 2 presents the FingerMouse system developed within this thesis and illustrates the use of such a device in different application scenarios. While these scenarios portray the motivation for the deployment of such a device, the scenarios also define the system requirements as well as the images we expect a FingerMouse system to encounter.
Chapter 3 shows the approach and algorithms we use to perform a foreground segmentation.

Chapter 4 evaluates our approach using data from the scenarios described in chapter 2.

Chapter 5 describes the hardware implementation of our different approaches.

Chapter 6 provides conclusions and an outlook.
We present a novel user input device for wearable computers: the FingerMouse. The device is an embedded stereo vision platform able to segment close objects against the background in the images the system captures. This segmentation is used to acquire the shape and position of a hand by measuring its complete area in the image, at a rate of 15-60 images per second.

We show two target applications of the device: hands free user input to wearable computers and a background removing camera for mobile video telephony.
2.1. System Description

2.1.1. Functionality of the System

The FingerMouse is a system that incorporates two cameras and a processing unit running vision algorithms. It captures a scene, as depicted in figure 2.1, at several images per second and transmits for each captured image the following outputs:

1. foreground / background segmentation map
   The system performs a foreground / background segmentation of the scene image captured by the right camera. The result is a segmentation map, a bitmap containing the classification of each image pixel.

   A pixel is classified as belonging to the foreground, when its range $Z$ falls into $[Z_{min}, Z_{max}]$. All other pixels are set to the background class. The range (or depth) of a pixel corresponds to its coordinate along the $Z$ axis of the scene object it belongs to. The $Z$ axis corresponds to the optical axis of the right camera (cf. figure 2.1). $Z_{min}$ and $Z_{max}$ can be configured to values from $Z_{prox}$ to $+\infty$.

   $Z_{prox}$ to $+\infty$ is the operating range of the system. Scene objects closer to the FingerMouse than the maximal proximity $Z_{prox}$ lead to undefined segmentation results.

   The FingerMouse transmits this segmentation map to a (wearable) computer, acting as a foreground segmenting smart camera. The map represents an acquisition of the silhouette of objects in the foreground and is used for further processing, depending on the application (cf. section 2.2).

2. position of the foreground
   The position of a foreground object is computed as absolute image pixel coordinates of its center-of-gravity $(x_{cog}, y_{cog})$ within the image captured by the right camera, as is shown in figure 2.2.

   The measurement of this position at several frames per second allows the position tracking over time of an object, e.g. the hand in the example of figure 2.1.
2.1. System Description

Figure 2.1: Capturing of a scene by the FingerMouse system: the system cameras are pointed along the Z axis. In this example the scene consists of a hand in the foreground, the range of the hand being contained between \(Z_{\text{min}}\) and \(Z_{\text{max}}\) (the space contained between the orange rectangles is the foreground). The background, the wall, is further away.

Figure 2.2: Segmentation map resulting from the scene shown in figure 2.1. White pixels are those classified to the foreground. The red cross marks the position of the center-of-gravity of the foreground \( (x_{\text{cog}}, y_{\text{cog}}) \).

3. scene depth image (disparity map)
Instead of the segmentation map output, the FingerMouse can also deliver a measurement of the depth of all the pixels in the form of four different disparity maps. These disparity measurements are further described in section 3.2.

4. raw camera images
The system can also output the color images (as those in figure 2.3) captured by its two internal cameras. This is useful for applications, that combine the original camera image with the segmentation results, such as in the example shown in figure 2.6 (p. 16). The combination of the right camera image and the segmentation map can be seen as a segmentation augmented image.

![Figure 2.3: Stereo images captured by the two cameras of the FingerMouse, according to the scene shown in figure 2.1.](image-url)
2.1. System Description

2.1.2. The FingerMouse Prototypes

The FingerMouse system uses a specifically designed ASIC performing the vision processing, which we denote the \textit{FingerMouse-IC}. Figure 2.4 shows a miniaturized hardware implementation of the FingerMouse system and table 2.1 shows the system specifications. More details related to the system hardware are presented in chapter 5.

![Figure 2.4: PCB and optics of miniaturized FingerMouse prototype, next to 2 Euro coin.](image)

<table>
<thead>
<tr>
<th>Specification</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>size</td>
<td>43 mm × 18 mm</td>
</tr>
<tr>
<td>power consumption</td>
<td>&lt; 200 mW</td>
</tr>
<tr>
<td>$Z_{\text{prox}}$</td>
<td>17 cm</td>
</tr>
<tr>
<td>field of view</td>
<td>$53^\circ \times 41^\circ$ (horizontal × vertical)</td>
</tr>
<tr>
<td>segmentation map resolution</td>
<td>320 × 480 pixels</td>
</tr>
<tr>
<td>image data rate</td>
<td>up to 60 frames/sec, 5 M pixels/sec</td>
</tr>
</tbody>
</table>

\textbf{Table 2.1: FingerMouse specifications}

**Nomenclature**

Prior to the ASIC-based FingerMouse prototype, two other generations of prototypes were implemented, one using a DSP as a processing core and the other using an FPGA.

We denote these systems \textit{DSP-FingerMouse} (cf. appendix A.7.1) and \textit{FPGA-FingerMouse} (cf. appendix A.7.2). This thesis is mainly focussed on the results from the ASIC-based FingerMouse, which we will simply denote \textit{FingerMouse} in the rest of this thesis.
2.2. Application Scenarios

In most vision processing that analyzes an object in a scene, the segmentation of the object of interest in the image is a crucial step. It is a standard task and many successful methods for static cameras and static backgrounds have been proposed over the past years, e.g. by Griesser et al. (2005, [27]). However if the system is in motion (e.g. when worn on the body), i.e. facing non-static backgrounds, this task becomes more challenging. Additionally, if power consumption, size or measurement latency are crucial, a device like the FingerMouse has to be used. The following scenario examples describe what the mobile foreground segmentation could serve for.

Section 2.2.1 shows the driving scenario and nomenclature origin of the FingerMouse. Section 2.2.2 shows another theoretical application scenario for the FingerMouse smart camera system.

2.2.1. Wearable Computing HCI

Worn on the body, the FingerMouse provides a user input interface to a wearable computer\(^1\). The system captures hand motion and gestures. No hardware or markers on the hand are needed, thus the system is a hands free input device (as shown in figure 2.5).

The segmented output can be used to recognize hand gestures (cf. Starner et al., 1997, [28]) or to extract other features, such as the orientation of the hand, the position of the fingertip, the hand silhouette, etc. A position tracking of the foreground area is already performed by the FingerMouse system.

Another application scenario is possible thanks to the FingerMouse system’s small size: instead of being worn as a separate device on the body, the system is integrated with a wearable, head-mounted display (HMD). The direction of the FingerMouse camera coincides with the display orientation, and the lens focal length is chosen, such that its field of view covers the field of view of the display. This setup allows the user to point directly into the virtual screen in front of him. In that case, a visual feedback, e.g. a mouse pointer, may not be necessary: the FingerMouse becomes a pointing device in its most direct form.

Static hand gestures can be recognized from a single segmentation map, dynamic hand gestures can be recognized from XY-tracking or

\(^1\)The computer shown in the figure 2.5 is the QBIC, cf. Amft et al. (2004, [7])
2.2. Application Scenarios

Figure 2.5: Wearable computing HCI scenario
The FingerMouse as an input device to wearable computers. The user moves his hand within a virtual plane, parallel to the FingerMouse device, in a similar fashion a computer mouse is used on a desk. In this example, the user controls a pointer in a virtual 2D computer image provided by a HMD. The area in which the hand can be moved is determined by the field of view of the device’s optics.

other features, analyzed over a time window. An overview of hand gesture recognition in the context of wearable computing is given by Moeslund et al. (2003, [29]) and Pavlovic et al. (1997, [30]).

2.2.2. Mobile Video Telephony with Background Removal
With the introduction of 3G mobile phones and networks, mobile video telephony (also called video conferencing with 2 users) has become available to any user of such a phone.

We propose a scenario where the FingerMouse system replaces the camera of such a phone, offering the possibility to remove or to blur the background in the telephony image, while conserving the image part showing the user (the foreground).

Such a background removal improves user privacy (hiding his current environment and location) and the privacy of other persons. Otherwise those persons could be filmed in the background during the video communication (cf. figure 2.6).
Figure 2.6: Mobile video telephony scenario
(a) the user is having a video telephony conversation, while a third person is present in the background.
(b) shows the unprocessed image as would be transmitted in standard video telephony
(c) shows a post-processed image, the background area detected by the FingerMouse has been blurred. The person in the background is now unrecognizable, and the data to be transmitted is less.

Furthermore, the bit rate of the video stream can be reduced (or its quality improved for a given bit rate): Since only the foreground (the person using the telephone) is the area of interest, the segmentation information allows to reduce the amount of information used in coding the background. An image with a removed or blurred background
2.2. Application Scenarios

will lead directly to such results. Furthermore, methods have been proposed to directly integrate the segmentation information into the video encoding process: Lin et al. (2003, [31]) present such a coding scheme using the H.263 coding standard\(^2\), and show that it "can significantly improve the PSNR\(^3\) and the subjective quality of face regions, while degradation introduced on the non-face areas is relatively invisible to human perception."

Another application for the segmentation and positioning information of the foreground in video telephony is the image stabilization. It reduces translations of the image that are caused by the user not holding the camera steadily (also called *jiggling*) by using a motion-compensated subwindow of the complete camera image. The stabilized image is more convenient to watch, and the encoding of the video stream is more efficient when less motion occurs. Such stabilization can use tracking information as the foreground position provided by the FingerMouse. Techniques for image stabilization in mobile image communication and the relation to encoding efficiency are described by Jachalsky et al. (2003, [33]) and by Engelsberg et al. (1999, [34]). The relevance of segmentation information in the motion estimation process is described by Wittebrood et al. (2001, [35]).

---

\(^2\)H.263 is a video codec originally designed by the ITU-T in 1995/1996 as a low-bitrate compressed encoding solution for videoconferencing, specified in [32].

\(^3\)The peak signal-to-noise ratio PSNR is often used to quantify video compression quality. The PSNR for image compression is the ratio between the mean difference (in the luminance channel) of the original and compressed images and the signal peak.
Algorithm for Background/Foreground Segmentation

In this chapter, we present an algorithm for foreground / background image segmentation. It is based on stereo vision depth mapping combined with color segmentation.

The algorithm is the base for the fast and power-efficient implementation of the FingerMouse-IC: four individual area-based correlations modules calculate ranges of pixels in the images, to be transformed into the binary domain of foreground/background classification. This allows the output stage to process pixels in boolean logic and to require less than 1000 bits of buffering.

At a second stage, a self-calibrating color segmentation is combined with the initial output, in order to enhance the segmentation for the case of uniformly colored objects, such as the human hand.

Finally, the algorithm also computes the position of the hand in the image.
3.1. Introduction: Algorithm Overview

This chapter describes the background / foreground segmentation algorithm implemented within the IC based FingerMouse device, presented in chapter 2.

The inputs to the algorithm (subsequently called FingerMouse algorithm) are two images concurrently captured by two cameras of the FingerMouse. The cameras are setup in parallel, as shown in figure 3.1, with a baseline \( b = 25 \text{mm} \). A geometrical model of this stereo camera setup is presented in appendix A.3.

![Figure 3.1: FingerMouse camera setup. The two camera are setup in parallel: their optical axes are parallel to the range axis Z.](image)

The resulting outputs of the algorithm are a foreground / background segmentation map, which is a binary classification of all pixels of the right camera image, and pixel coordinates of the center-of-gravity of the foreground area in the right camera image.

In the following text, we provide an overview of the algorithm. The processing of the stereo input images is performed in several sequential steps, as shown in figure 3.2:

- depth mapping

  This algorithm layer uses four different modules, each performing stereo block matching on the grey-level intensity values of the stereo images. For the parallel camera setup, a disparity value for each pixel can be calculated, which determines the depth and real world coordinates relative to the camera of that pixel (cf.
appendix A.3). Two $L \gg R$ modules calculate the depth of pixels in the left camera image, the two $R \gg L$ modules calculate the depth of the right camera image pixels. Each of the two modules is based on a different correlation metric, called census and SAD. (Cf. section 3.2.)

- perspective transform of $L \gg R$ results

The disparity values measured by the $L \gg R$ depth mappings apply to the pixels in the image captured by the left camera. Those measurements are converted so that they describe depth values for the pixels of the right camera image. (Cf. section 3.4.1.)

- disparity thresholding
In this step, the disparity values measured in the preceding steps are converted to a binary value, representing the pixel’s membership of either the foreground or background class. The result is a segmentation map, yielding the class of every pixel. (Cf. section 3.3.)

- segmentation map fusion
  The preceding step generates four different segmentation maps, one for each depth mapping module. The fusion of the four maps leads to a single segmentation map with a lower false classification rate. (Cf. section 3.4.2.)

- noise filtering
  This step further eliminates false measurements in the segmentation map output. The result of this step is the $S_{\text{filtered}}$ segmentation map, one of the outputs of the algorithm, called segmentation map 1 in figure 3.2. (Cf. section 3.5.)

- optional area filtering with color segmentation
  In this optional step, the segmentation map produced in the preceding step can be further optimized, using an analysis of the color information of the foreground class. The resulting segmentation map is the $S_{\text{color-\ combined}}$ algorithm output, called segmentation map 2 in figure 3.2. (Cf. section 3.6.)

- position tracking
  This step calculates the center-of-gravity coordinates of the foreground area in the segmentation map. In case of a single object in the foreground (e.g. the user’s hand), this measurement reveals the position of the center of the object within the right camera image. (Cf. section 3.7.)

3.2. Depth Mapping Step

When a scene is captured by two separate cameras and the relative position of the cameras is known, the three dimensional composition of the scene can be calculated. For a spot that is visible in both camera images, the coordinates of the spot in the two images allows to triangulate its position in real-world coordinates, relative to the camera position (cf. figure 3.1).
The geometry of binocular stereo vision and the triangulation principle is widely covered by literature (e.g. by the books [36] and [37] by Hartley, Zissermann and Faugeras). For the case of a parallel camera setup such as in the FingerMouse shown in figure 3.1, the relative positions of a spot in the two stereo images always correspond to a translation along the horizontal image line. A geometrical demonstration is given in appendix A.3. The spot is translated to the left, in the right camera image, and vice versa. The amount of this translation, measured in pixels, is the so-called disparity $d$, and it is inversely proportional to $z$, the depth or range of the spot, as described by equation 3.1:

$$z = \left( b \cdot f \cdot \frac{X_i}{x_s} \right) \cdot \frac{1}{d}$$  (3.1)

For later reference, we define the factor $K$:

$$K = \left( b \cdot f \cdot \frac{X_i}{x_s} \right)$$  (3.2)

The depth or range $z$ corresponds to the spot coordinate along the axis $Z$, as shown in figure 3.1. The baseline $b$ is the translational distance of the cameras (cf. figure 3.1), $f$ is the focal length of the lenses, $X_i$ is the horizontal image resolution in pixels and $x_s$ is the width of the image sensor’s sensitive area. The FingerMouse system presented in chapter 2 uses the following values:

$$b = 25 \, \text{mm}, \quad f = 3.6 \, \text{mm}, \quad X_i = 320 \, \text{pixels}, \quad x_s = 3.6 \, \text{mm}.$$  (3.3)

Figure 3.3 shows an example of a stereo set captured by a setup of parallel cameras, and makes the horizontal disparities and their relation to the depth structure of the scene visible. This stereo set will be used as an example to demonstrate the function of the depth mapping layer.

### 3.2.1. Specification of the Implemented Area-Based Matching

The fundamental problem for the disparity determination is the identification of the two corresponding or matching pixels that describe the same spot in the two images. Our approach belongs to the class of area-based matching algorithms, which are suitable for real-time implementation (e.g Faugeras et al., 1993, [39]). The choice of the parallel
Figure 3.3: Sample of a stereo image set, captured by two parallel cameras. This set, called after its origin Tsubuka, is commonly used in literature as a reference set to compare disparity mapping algorithms, it was introduced by Nakamura et al. (1996, [38]).

(a) and (b) show the two camera images (c) shows an overlay of them. It is visible that close objects, like the lamp, are shifted horizontally by a bigger distance.

(d) shows a ground truth disparity map, which indicates the true disparity of objects of the left camera image. We denote \( D_{gt,L}(p) \) and \( D_{gt,R}(p) \) the functions returning the true disparity of a pixel \( p \) in the left and right camera images \( I_l \) and \( I_r \) respectively.

(e) is the scale for the color mapping of the disparity values.

camera setup leads to a one dimensional search space parallel to the horizontal image lines is also key to the efficiency, as no interpolation of pixel values is needed (cf. Barnard et al. (1982, [40])).

The FingerMouse algorithm performs a correspondence search for each pixel in the right camera image \( I_r \) by two different block matching modules, each of them producing a disparity measurement. This corre-
spondence search direction is denoted $R \gg L$. The same search with 2 modules is done for pixels in the left camera image $I_l$, and is denoted $L \gg R$ search.

As illustrated in figure 3.4, the $R \gg L$ correspondence search for a pixel $p_1$ of pixel coordinates $(x_1, y_1)$ in $I_r$ is done by comparing it to candidate pixels in $I_l$. The candidate pixels are all the pixels in $I_l$ of coordinates $(x_1 + t, y_1)$, for all $t$, $0 \gg t \gg d_{\text{limit}}$. The value $d_{\text{limit}}$ represents a chosen limit to the search space size (and the related computational effort), but also represents the highest disparity that can be measured.

The comparison of $p_1$ and a candidate pixel $p_{\text{candidate}}$ considers a window of size $(2H+1) \times (2W+1)$ around the pixels, the so called blocks. The window around $p_1$ is called reference window, the overlapping windows around the candidate blocks constitute the search window. The block with the highest similarity to the reference window is chosen as the match guess, its offset $t_0$ is the measured disparity.

![Figure 3.4: Reference and search window for the $R \gg L$ disparity computation of pixel $p_1(x_1, y_1)$. The pixel $p'_1$ represents the corresponding pixel for $p_1$, and is located at $(x_1 + t_0, y_1)$.

The FingerMouse ASIC performs the described block matching with a block size of $3 \times 5$ pixels, $(H = 1, W = 2)$ and a disparity search width of $d_{\text{limit}} = 47$ pixels. Along with the FingerMouse camera parameters from equation 3.3, the maximum disparity $d_{\text{limit}}$ corresponds to a depth of 17.02 cm at a horizontal camera resolution of 320 pixels, using equation 3.1. This depth, called $Z_{\text{prox}}$, represents the minimum range for objects, that the algorithm can still match.
In order to compare candidate blocks to the reference window, each module uses a specific correlation function \( C(t) \), expressing the similarity to candidate blocks of an offset \( t \). One block matching module uses a \textit{SAD-based} (sum of absolute differences) correlation function \( C_{SAD}(t) \), the other module uses a \textit{census-based} correlation function \( C_{census}(t) \).

The function \( C_{SAD,R \gg L}(t) \) for the pixel \( p_1(x_1, y_1) \), \( R \gg L \) search, is defined as follows:

\[
C_{SAD,R \gg L}(t, (x_1, y_1)) = \\
\sum_{dx=-W}^{+W} \sum_{dy=-H}^{+H} |I_r(x_1 + dx, y_1 + dy) - I_l(x_1 + t + dx, y_1 + dy)|
\]

\( I_r(x, y) \) and \( I_l(x, y) \) are the intensity (brightness) values of pixels located at \( (x, y) \) in \( I_r \) and \( I_l \) respectively. The value of the function \( C_{SAD}(t, (x_1, y_1)) \) expresses the difference between the reference window and a candidate block of offset \( t \): the best matching candidate block minimizes the function, and the offset \( t_0 \) at the minimum represents the disparity measurement. If several equal minima occur, the one at the smallest offset is chosen for \( t_0 \). We denote \( D_{SAD,R \gg L}(p_1) \) the function measuring the disparity \( t_0 \) for a pixel \( p_1 \) using the \( SAD, R \gg L \) module.

Figure 3.5 shows the \( C_{SAD,R \gg L} \) function applied to a pixel \( p_1 \) from the Tsubuka stereo set, using the matching parameters of the FingerMouse (\( H = 1, W = 2, d_{\text{limit}} = 47 \)).

The second \( R \gg L \) module uses the \( C_{census} \) correlation function, based on the \textit{census transform} \( \Phi \).

The \textit{census} transform \( \Phi \) maps blocks to a bit-sequence: a \( 3 \times 3 \) pixel block (\( W = 1, H = 1 \)) is mapped to 8 bits, every bit representing whether a boundary pixel has a higher or lower intensity than the center pixel. The \textit{census} transform of a \( 5 \times 3 \) pixel block (\( W = 2, H = 1 \), the block size in the FingerMouse implementation) is the concatenation of the \textit{census} transforms of the three \( 3 \times 3 \) blocks it contains (24-bit sequence). In order to compare blocks, the hamming distance\(^1\) \( \Delta_{\text{Hamming}} \) between their \textit{census} transforms is computed, and the function \( C_{census,R \gg L}(t) \) for the pixel \( p_1(x_1, y_1) \), \( R \gg L \) search, is defined as follows:

\(^1\)“In information theory, the \textit{Hamming distance} between two strings of equal length is the number of positions for which the corresponding symbols are different (…)” (Hamming et al., 1950, [41])
3.2. Depth Mapping Step

$$C_{\text{SAD}}(t, (x, y))$$

The $$R \gg L$$, SAD correlation function $$C_{\text{SAD}, R \gg L}(t, (x_1, y_1))$$ for pixel $$p_1$$ ($$x_1 = 134$$, $$y_1 = 191$$, located at the nose of the head), from the Tsubuka stereo image. It minimizes at $$t_0 = 22$$, which corresponds to the correct disparity, as can be seen in the enlarged image subwindows. $$p'_1$$ is the corresponding pixel for $$p_1$$.

$$C_{\text{census}, R \gg L}(t, (x_1, y_1)) =$$

$$\Delta_{\text{Hamming}}(\Phi(\text{block}_r(x_1, y_1)), \Phi(\text{block}_l(x_1 + t, y_1)))$$

$$\Phi(\text{block}_{r/l}(x, y))$$ represents the census transform of a block (sized 5 × 3 pixels in this implementation) centered around $$(x, y)$$ in the $$I_r/I_l$$ respectively. The disparity measured by the census, $$R \gg L$$ module for pixel $$p_1(x_1, y_1)$$ equals the offset $$t_0$$ that minimizes $$C_{\text{census}}$$, similar to the SAD, $$R \gg L$$ module. We denote $$D_{\text{census}, R \gg L}(p_1)$$ the function returning the measured disparity $$t_0$$ for a pixel $$p_1$$.

Two more $$L \gg R$$ block matching modules operate in a similar way also using SAD and census based correlation. They produce disparity measurements for all the pixels in $$I_l$$. Due to the stereo image
geometry, the search window in $I_r$ is now placed left of the $t = 0$ candidate, and the offsets $t$ measure an offset to the left. The correlation functions $C_{SAD,L\gg R}$ and $C_{census,L\gg R}$ for the $L \gg R$ block matching differ accordingly. We denote $D_{SAD,L\gg R}$ and $D_{census,L\gg R}$ the functions returning the disparities measured by the $L \gg R$ block matching modules.

**Results**

Figure 3.6 shows the resulting disparities for all pixels measured by the four disparity measuring functions $D_{SAD,R\gg L}$, $D_{census,R\gg L}$, $D_{SAD,L\gg R}$ and $D_{census,L\gg R}$ of the FingerMouse algorithm. This graphical representation is called disparity map or depth map, as disparity and depth are directly related by formula 3.1.

![Disparity Maps](image)

Figure 3.6: Results A: The four depth maps (a-d) displaying the disparities measured by the four individual block matching modules of the FingerMouse. (e) is the scale for the color mapping of the disparity values, which also represent depth values according to formula 3.1.

An evaluation of disparity measuring algorithms including quality
metrics is presented in the work of Scharstein et al. (2002, [42]). We apply two of Scharstein’s quality measures to the results computed by the FingerMouse algorithm for the Tsubuka stereo sample:

- The RMS error $R$ (measured in disparity units, i.e. pixels) between disparity calculations $D(p)$ and the ground truth disparities $D_{gt}(p)$ (cf. figure 3.3(d)):

$$R = \left( \frac{1}{N} \sum_{\forall p} |D(p) - D_{gt}(p)|^2 \right)^{\frac{1}{2}}$$ (3.6)

where $N$ is the number of pixels.

- The percentage of bad matching pixels $B$:

$$B = \frac{1}{N} \sum_{\forall p} (|D(p) - D_{gt}(p)| > \delta_d)$$ (3.7)

where $\delta_d$ is a disparity error tolerance.

Comparisons of other stereo block matching algorithms using the quality measures $R$ and $B$ and the Tsubuka stereo sample have been published by Szeliski et al. (1999, [43]); Kolmogorov et al. (2001, [44]) and Scharstein et al. (2002, [42]).

Table 3.1: RMS error $R$ and bad pixel percentage $B$ for Results A of the Tsubuka stereo sample, $L \gg R$ (we only have ground truth data for the left camera image)

<table>
<thead>
<tr>
<th></th>
<th>$R$</th>
<th>$B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>ground truth</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$D_{SAD,L \gg R}$</td>
<td>6.93 pixels</td>
<td>20.3%</td>
</tr>
<tr>
<td>$D_{census,L \gg R}$</td>
<td>10.77 pixels</td>
<td>34.1%</td>
</tr>
</tbody>
</table>

Discussion

The results in figure 3.6 and in table 3.1 show that most of the disparities are measured correctly (79.7 % and 65.9 % respectively), but a high number of false measurements still occur, the census module producing more errors.
Chapter 3: Algorithm for Background/Foreground Segmentation

On the other hand, the four measured disparities represent an intermediary result, which is further processed by fusion and filtering. The foreground/background segmentation calculations performed on the segmentation processing layer show a much smaller percentage $B$ of bad pixels, but the measurement is conducted in the binary segmentation domain, and therefore it is not comparable anymore to $B$ values of disparity measuring algorithms.

Although the census based block matching performs worse than the $SAD$ based one, the $D_{\text{census}}$ measurements are useful to be combined with the $D_{\text{SAD}}$ measurements, as will be done in the fusion step. As a non-parametric approach, the census block matching (cf. Banks et al., 1997, [45]; Zabih et al., 1994, [46]) considers only relative intensities of pixels within the block, and thus is invariant to a bias of the intensity of the respective input images. An intensity bias occurs as lenses (especially miniature lenses as in the FingerMouse system) have a non uniform transmission over the area (usually the center is most transmissive), and through radiometric differences between the two image sensors. The different characteristics of the $SAD$ and census based measurements help retrieve more foreground pixels in the segmentation stage.

The disparities measured by other algorithms, e.g. those shown in [42], perform better in terms of $R$ and $B$, since they all use methods of post-processing the disparity measurements, e.g. the interpolation of uncertain measurements (cf. Fua et al., 1991, [47]). The post-processing of disparity values is more costly in terms of processing and data storing than the segmentation layer we propose, and most algorithms presented in [42] are not optimized for execution speed.

3.2.2. Characterization of False Depth Measurements

When comparing the resulting disparity measurements of the four different implemented modules (figure 3.6) to the true disparity values of the scene (shown in the ground truth disparity map, figure 3.3 (d)), it is visible that the method produces some false measurements. The location and distribution of these false measurements differs for each module.

There are several causes for these errors, inherent to the block matching approach:

- In regions of uniform pixel intensity distribution (e.g. around $p_2$ in the Tsubuka scene, cf. figure 3.7a) the minimum of the cor-
3.2. Depth Mapping Step

Figure 3.7: (a) The pixels \(p_2, p_3, p_4\) are examples of pixels placed in different regions of the Tsubuka sample (left camera image shown) where depth calculations are often erroneous, as discussed in section 3.2.2. (b) shows all the pixels (in white) for which the \(SAD_L \gg R\) module produces a disparity measurement error larger than four pixels:

\[ |D_{SAD,L} - D_{gt,L}| > 4 \]

relation functions (equations 3.4 and 3.5) is not well defined, as neighbouring candidate blocks are identical, or almost. This problem is even stronger when using the census based correlation, as it has a smaller result range ([0..24] for \(5 \times 3\) blocks). While our implementation picks the smallest \(t\) when several minima of \(C(t)\) exist, the problem is still persistent, since even blocks at the correct offset are almost never identical to the reference block.

- In regions containing horizontally periodical structures, the correlation function produces additional (erratic) minima, as candidate blocks offset by the horizontal period can be confused with the correct matching block. The pixel \(p_3\) is an example of a pixel contained in such a region. The horizontal period (the video cassettes in the shelve) equals 10 pixels, leading to some measurements of \(t_0 = 18, t_0 = 28\) while the correct disparity equals 8.

- In areas near outer borders of objects in the foreground, the corresponding pixel is often occluded by an object of smaller range (occlusions). This is the case for example for pixel \(p_4\): the same area is not visible in the right camera image, as it is occluded by the lamp in that perspective. Hence, no matching pixel can be
found, and the result of the measurement is erroneous. The block matching modules will output a calculation of the most similar candidate block. Occlusions and their handling in stereo vision is described by Chang et al. (1991, [48]); Egnal et al. (2002, [49]).

- At the left camera image border (for \( L \gg R \) search, and vice versa), the disparities cannot be computed, since the corresponding pixels for far objects fall outside the frame of the right camera image. In literature, the left and right image border area is most often simply omitted and its pixels not counted in evaluations (e.g. in [42]). We will do the same in our evaluations.

Since the affected area is deterministic, the effect does not induce noise but rather reduces the practical field of view. A measurement in that area is still possible, but the result will be false if the matching pixel falls out of the image. In the FingerMouse implementation, this image area is processed. The search window is reduced to fit near the borders. The result is a higher noise rate in the border area.

- Objects with a non-planar shape characteristic (the object surface seen by the camera is not parallel to the image plane) appear differently within the two camera perspectives, leading to a diminished block similarity for corresponding pixels. The effect decreases when choosing a smaller baseline \( b \) and focal length \( f \), or when the object has a front-to-parallel position towards the system, e.g. a hand palm facing the camera.

### 3.3. Foreground Segmentation

The measured disparities \( t_0 \) from the four different block matching modules are converted to a binary value by the segmentation function \( S \). This value expresses a pixel’s classification to either the foreground (1) or background (0) class.

The segmentation function \( S \) performs a range clipping on the measured disparity, and is defined as follows:

\[
S : [0, d_{\text{limit}}] \rightarrow \{0, 1\}
\]

\[
S(t_0) = 1, \text{ for all } t_0, d_{\text{min}} < t_0 < d_{\text{max}}
\]

\[
S(t_0) = 0, \text{ for all } t_0, t_0 \leq d_{\text{min}} \text{ or } t_0 \geq d_{\text{max}}
\]
3.3. Foreground Segmentation

$d_{\text{min}}$ and $d_{\text{max}}$ are two threshold variables that can be set to the system. $d_{\text{min}}$ corresponds to the threshold distances $Z_{\text{max}}$ (introduced in section 2.1, p. 10), $d_{\text{max}}$ expresses the close range threshold $Z_{\text{min}}$.

The segmentation map is the graphical representation of the function $S$ applied to the disparity measuring functions, showing the classification result for all pixels. Figure 3.8(c) shows the segmentation map for $S(D_{\text{SAD,L} \gg R})$, where the segmentation function $S$ was applied to the output of the $\text{SAD,L} \gg R$ module, using the Tsubuka sample and threshold values $(d_{\text{min}}, d_{\text{max}}) = (26, 36)$, which cover the disparity range of the lamp.

Table 3.2 shows how we denominate the correctness of a segmentation result, taking as a reference the ground truth results. Figure 3.8(d) shows the location of false classifications in the $S(D_{\text{SAD,L} \gg R})$ measurement.

**Table 3.2:** Denomination of classification correctness, for a pixel $p$ classified by $S(D_{\text{SAD,L} \gg R})$.

<table>
<thead>
<tr>
<th>$S(D_{\text{SAD,L} \gg R}(p)) = 0$</th>
<th>$S(D_{\text{SAD,L} \gg R}(p)) = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S(D_{\text{gt,L}}(p)) = 0$</td>
<td>true negative, $t_n$</td>
</tr>
<tr>
<td>$S(D_{\text{gt,L}}(p)) = 1$</td>
<td>false positive, $f_p$</td>
</tr>
<tr>
<td>$S(D_{\text{gt,L}}(p)) = 0$</td>
<td>false negative, $f_n$</td>
</tr>
<tr>
<td>$S(D_{\text{gt,L}}(p)) = 1$</td>
<td>true positive, $t_p$</td>
</tr>
</tbody>
</table>

Migration of False Disparity Measurements to False Segmentation Classifications

False classification results are caused by false disparity measurements (described in section 3.2.2), but not all false disparity measurements lead to a false classification:

- False disparity measurements resulting of occlusion lead to false positives around objects of smaller range, but they do not lead to false negatives, as can be seen by comparing figures 3.8(b) and 3.8(d). (In that example, the false disparity measurements due to occlusion are visible at the left border of both the sculpture and the lamp.) The reason is that the lamp, being the closest object in the scene, is not occluded by any other object. This assumption is also valid for the FingerMouse application scenarios we present in section 2.2, where the hand or person is also the closest object in the captured scene.
Figure 3.8: Foreground segmentation through range clipping from the SAD L > R disparity map, \((d_{\min}, d_{\max}) = (26, 36)\)

(a) ground truth segmentation map, derived from the ground truth disparity function \(D_{gt,L}\). Pixels \(p\) classified to foreground \((S(D_{gt,L}(p)) = 1)\) are drawn in white, pixels classified to background in black.

(b) shows all the pixels (in white) for which the SAD L > R module produces a disparity measurement error larger than four pixels: \(|D_{SAD,L\gg R} - D_{gt,L}| > 4\) (This is a repetition of figure 3.7(b))

(c) segmentation map from thresholded SAD L > R disparity/depth map. Pixels \(p\) classified to foreground \((S(D_{SAD,L\gg R}(p)) = 1)\) are drawn in white, pixels classified to background in black.

(d) same as (c), but with false positives \(f_p\) colored blue, false negatives \(f_n\) colored red \((f_p, f_n\) are defined in table 3.2).

- False positives occur when the disparity measurements in the background are false, and the false result falls into the foreground window defined by \((d_{\min}, d_{\max})\) (cf. figure 3.9). Most of the false disparity measurements do not generate a false positive, since the
probability is high that the false measurement is closer to the correct disparity (as shown in figure 3.10 for the Tsubuka sample) and is not contained in the foreground window. The reason for the disparity error $\Delta = |D_{SAD,L} \gg R - D_{gt,L}|$ tending to be small, lies in the nature of a scene: if the correct matching block is missed, another similar block is likely to be close rather than far away.

- False negatives are caused by false disparity measurements within the foreground region, when the resulting disparity falls out of the foreground window defined by $(d_{min}, d_{max})$.

![Figure 3.9: Distribution of false disparity measurements ($\Delta = |D_{SAD,L} \gg R - D_{gt,L}| > 4$) for all pixels belonging to the ground truth background (black pixels in figure 3.8(a)) of the Tsubuka sample. All $D_{SAD,L} \gg R(p)$ disparity calculations resulting in 26 to 36 generate false positives (blue pixels in figure 3.8(d)) after the thresholding.](image)
Figure 3.10: Distribution of the relative disparity error $\Delta = |D_{SAD,L,R} - D_{gt,L}|$, for erroneous pixels ($\Delta > 2$) within the ground truth background (black pixels in figure 3.8(a)) of the Tsubuka image set.

3.4. Fusion to a Single Segmentation Map

This section describes how the disparity measurements from the four different modules are transformed and combined into a single segmentation result for each pixel in the right camera image perspective. Before the fusion operation (cf. section 3.4.2) is carried out, the outputs from the $L \gg R$ modules are transformed to match pixels in the right camera perspective, as shown in the following section (3.4.1).

3.4.1. Left-Right Perspective Conversion of Disparity Measurements

The results from the four disparity measurement modules apply to either pixels of the right ($R \gg L$ modules) or left ($L \gg R$ modules) camera image. However, the disparity measurements of a pixel and its horizontal neighbors within a distance of $d_{limit}$ contain the information necessary to transform the results, so that they apply to the complementary perspective.

We define the perspective shift transformation $P_{shift}$, as a transformation that remaps the measurements of a module to pixels in the complementary image.

For each $R \gg L$ result $t_0$ of a pixel $p_{right}(x_{right}, y_{right})$ in $I_r$, the result $t_0$ is assigned to a pixel $p_{left}$ in $I_l$, of the following coordinates $(x_{left}, y_{left})$: 

$$

x_{left} = x_{right} + P_{shift}(x_{right}) \\
y_{left} = y_{right} - P_{shift}(x_{right})

$$

$$

P_{shift}(x) = \left\{ \begin{array}{ll}
- \frac{d_{limit}}{2} & \text{if } x < -\frac{d_{limit}}{2} \\
0 & \text{if } -\frac{d_{limit}}{2} \leq x \leq \frac{d_{limit}}{2} \\
\frac{d_{limit}}{2} & \text{if } x > \frac{d_{limit}}{2} 
\end{array} \right.

$$
3.4. Fusion to a Single Segmentation Map

\[ x_{left} = x_{right} + t_0 \quad y_{left} = y_{right} \] (3.9)

This corresponds to a shift to the right, by the distance of the disparity measured. In the case of a transformation of \( L \gg R \) results to the right camera perspective, the shift occurs to the left.

The transformation \( P_{shift} \) is non-injective: it can occur that several measurements are remapped to the same pixel coordinates. In that case, the highest disparity measurement is applied. Furthermore, some pixels in the new perspective are not assigned to a disparity value. We define their disparity to be 0.

Figure 3.11 shows how \( P_{shift} \) transforms the results for the Tsubuka sample calculated by \( D_{SAD,R \gg L} \) to the left camera perspective.

**Figure 3.11:** Example of a perspective transformation by \( P_{shift} \), from right to left camera viewpoint

(a): disparity map, by \( D_{SAD,R \gg L} \), from the Tsubuka sample (identical to fig. 3.6(c))

(b): The same measurements as in (a), after transformation by \( P_{shift} \) to the left camera perspective. White pixels are those not remapped, which are set to disparity 0.

(c) is the scale for the color mapping of the disparity values.

3.4.2. Fusion

After shifting the disparity results from the \( L \gg R \) modules to the right camera perspective, four segmentation results are generated through the segmentation function \( S \). We can write those four segmentation results for a pixel \( p \):
Chapter 3: Algorithm for Background/Foreground Segmentation

\[ S_{SR}(p) = S(D_{SAD,R \gg L}(p)) \]
\[ S_{CR}(p) = S(D_{census,R \gg L}(p)) \]
\[ S_{SL}(p) = S(P_{shift}(D_{SAD,L \gg R}(p))) \]
\[ S_{CL} = S(P_{shift}(D_{census,L \gg R,shift}(p))) \]

Figure 3.12 shows a new sample scene on which we demonstrate the segmentation fusion algorithm. The scene was captured with the same camera baseline \((b = 25 \text{ mm})\) and optical configuration \((f = x_s)\) as is used in the FingerMouse prototype. In order to perform a foreground segmentation of the hand \((z = 47 \text{ cm})\), the segmentation function \(S\) operates with a range window of: \((d_{\text{min}}, d_{\text{max}}) = (15, 21)\), which corresponds to a depth range of \((Z_{\text{min}}, Z_{\text{max}}) = (38 \text{ cm}, 53 \text{ cm})\). Figure 3.14 shows the segmentation results calculated by the four different modules. Figure 3.13 shows four depth maps calculated by the depth mapping layer.

As can be seen in figure 3.14, the four different segmentation functions produce recognizable segmentation of the hand against the background, but also generate a number of false positives and false negatives. The actual count of false classifications is presented in table 3.3 (p. 51). The sources of those false measurements are inherent to the different block matching techniques described in section 3.2.2 and 3.3. As has been described as well, the location and distribution of falsely classified pixels differ for each module.

By fusing the four functions into a single one, we want to reduce the amount of false classifications. The first step is to combine the results of the \(SAD\) and \(census\) based functions via a logical \(OR\) operation, for both \(R \gg L\) and \(L \gg R\) results: all pixels classified to \(foreground\) (logical 1) by either the \(SAD\) or the \(census\) module (or both) are now classified to \(foreground\). In figure 3.15, the fused results \((S_{SR} OR S_{CR})\) and \((S_{SL} OR S_{CL})\) can be seen: compared to the outputs of the single modules (figure 3.14), the results now yield less false negatives (cf. table 3.3, p. 51) as the \(SAD\) and \(census\) results recognize different parts of the foreground correctly. The amount of false positives (pixels in the background falsely classified to \(foreground\)) rises, as false positives from both the \(SAD\) and \(census\) measurements result in false positives in the fused results.

In a second step, the two \(OR\)-combined \(R \gg L\) and \(L \gg R\) results are merged into a final result, by a logical \(AND\) operation: Pixels will
3.4. Fusion to a Single Segmentation Map

be classified to foreground only if both \((S_{SR} OR S_{CR})\) and \((S_{SL} OR S_{CL})\) results are logically positive (foreground).

As can be seen in figure 3.16 and in table 3.3, the resulting function \(((S_{SR} OR S_{CR}) AND (S_{SL} OR S_{CL}))\) produces less false positives. Occlusion errors are eliminated, as they occur individually at either the left or right side of foreground objects: the AND-combination performs a left-right consistency check.

We denote the final result \(S_{combined}\), it can be defined from the four segmentation functions:

\[
S_{combined} = (S_{SR} OR S_{CR}) \ AND \ (S_{SL} OR S_{CL})
\]  

(3.10)
(a) module: census, $R \gg L$

(b) module: census, $L \gg R$

(c) module: SAD, $R \gg L$

(d) module: SAD, $L \gg R$

(e)

Figure 3.13: Results A2 for the model hand stereo set: the four depth maps (a-d) displaying the disparities measured by the four individual block matching modules of the FingerMouse. (e) is a scale for the colour mapping of the disparity values, which also represent depth values according to formula 3.1.

Conclusion

The fusion operation is performed as a boolean operation, combining the outputs from the block matching modules $(S_{SL}(p), S_{SR}(p), S_{CL}(p), S_{CR}(p))$ directly to a single binary value $S_{combined}(p)$ for each pixel $p$, thus reducing the data for the subsequent processing to 1 bit / pixel.

The processing of pixels in the segmentation processing layer, such as the occlusion elimination via an AND operator, relies on simple boolean functions, which enables efficient CMOS circuit implementation.

An important factor that makes this boolean occlusion elimination possible is the absence of occlusions within the foreground area: according to the scenarios we presented, the hand or person to be segmented is not occluded, as it is the closest object in the scene. Hence, classify-
3.5. Noise Filtering

In order to reduce the amount of false positive measurements in the output of $S_{combined}$, the result is processed by a noise filter. The filter operates on a pixel’s classification bit $S_{combined}(p)$, and on the classification of four of its neighbors (two on the left, two on the right), and produces a new classification result, $S_{filtered}(p)$. The filter performs an erosion with a blocksize of $5 \times 1$. For a pixel of coordinates $(x_1, y_1)$, the value of the filtered result $S_{filtered}(x_1, y_1)$ can be expressed as a logical function, as defined in equation 3.11.

**Figure 3.14:** Segmentation maps for right camera perspective, calculated by the four different block matching modules, using a depth range of $(Z_{min}, Z_{max}) = (38 \text{ cm}, 53 \text{ cm})$

Coloring: true positive: white, true negative: black, false positive: yellow, false negative: red

ing all left-right mismatching results to background via AND results in true negative $t_n$ classifications.
Figure 3.15: (a) segmentation results for the function $(S_{SL} \ OR \ S_{CL})$, the left-right combination of the two SAD-modules. (b) segmentation results for the function $(S_{SR} \ OR \ S_{CR})$, the left-right combination of the two census-modules.

Coloring: true positive: white, true negative: black, false positive: yellow, false negative: red

The top line shows the four segmentation results from figure 3.14, without coloring (white = pixels classified to foreground, left to right: $S_{SL}$, $S_{CL}$, $S_{SR}$, $S_{CR}$)

\[
S_{filtered}(x_1, y_1) = S_{combined}(x_1 - 2, y_1) \\
AND S_{combined}(x_1 - 1, y_1) \\
AND S_{combined}(x_1, y_1) \\
AND S_{combined}(x_1 + 1, y_1) \\
AND S_{combined}(x_1 + 2, y_1) \tag{3.11}
\]
3.5. Noise Filtering

Figure 3.16: (a) segmentation results for the function $S_{combined} = (S_{SR} \lor S_{CR}) \land (S_{SL} \lor S_{CL})$
Coloring: true positive: white, true negative: black, false positive: yellow, false negative: red
The top line shows the two segmentation results from figure 3.15, without coloring (white = pixels classified to foreground, left to right: $(S_{SR} \lor S_{CR}), (S_{SL} \lor S_{CL})$ )

For pixels $(x,y)$ falling out of the image border, we assume $S_{combined}(x,y) = positive$.

Results

Figure 3.17 shows the result of the noise filter applied to the $S_{combined}$ result. The noise filtering leads to the classification of some new pixels to background: the filtered result $S_{filtered}$ shows less false positive classifications, but an increased number of false negatives (cf. table 3.3). As seen in figure 3.17 and in actual numbers in table 3.3, very few
false positives remain (26 $f_p$ pixels out of 68480 image pixels). This means, the pixels classified as *foreground* / positive, are very likely to be true positives: the fusion and filtering lead to a robust removal of the background and a sparse recognition of the foreground. A subsequent *dilation* with the same block size $5 \times 1$ can reverse foreground pixels lost in the *erosion*, but is not carried out within the algorithm.

The resulting classification $S_{filtered}$ is the first of two segmentation outputs of the FingerMouse algorithm, we call it *Result B*. A second output uses a color segmentation module, which is fused with with $S_{filtered}$ in order to reduce the number of false negatives, i.e. to make the recognition of the foreground more dense (cf. section 3.6).
3.6. Fusion with Color Segmentation Module

The color segmentation module is designed to enhance the segmentation measurements from the $S_{\text{filtered}}$ filter output (figure 3.18(a)). It

Figure 3.17: (a) segmentation results for the function $S_{\text{combined}}$, as displayed in figure 3.16. (white = pixels classified to foreground) (b) Result B: the segmentation results for $S_{\text{filtered}}$, resulting of the noise filtering of $S_{\text{combined}}$. Coloring: true positive: white, true negative: black, false positive: yellow, false negative: red
uses color information of the right camera image (figure 3.18(b)) to reclassify pixel as foreground / positive that have a similar color than the foreground pixels determined by $S_{\text{filtered}}$ and that are within a defined proximity of the detected foreground area.

Figure 3.18: Inputs to the color segmentation module:
(a) segmentation results for the function $S_{\text{filtered}}$, as displayed in figure 3.16. (white = pixels classified to foreground)
(b) greyscale representation of the hue values $\text{hue}(p)$ of each pixel. (black=0, white=255).

Figure 3.19: Structure of the color segmentation module

The color segmentation module consists of two components (cf. figure 3.19):
3.6. Fusion with Color Segmentation Module

- The *average hue computation component* measures the average hue mean of all hue values hue\( (p) \) of pixels \( p \) classified by \( S_{\text{filtered}} \) as foreground.

\[
hue_{\text{mean}} = \frac{1}{N_{\text{filtered}}(\text{foreground})} \cdot \sum_{\forall p, S_{\text{filtered}}(p) = 1} \text{hue}(p) \tag{3.12}
\]

\( N_{\text{filtered}}(\text{foreground}) \) is the number of pixels classified by \( S_{\text{filtered}} \) to foreground:

\[
N_{\text{filtered}}(\text{foreground}) = \sum_{\forall p, S_{\text{filtered}}(p) = 1} 1 \tag{3.13}
\]

The \( \text{hue}(p) \) value is an 8-bit value, describing the color tone (e.g. red, yellow...) of a pixel \( p \), as can be seen in figure 3.18(b). A definition of \( \text{hue} \) and the relation between RGB and HSV color space is presented by Vezhnevets et al.\((2003,[50])\). The practical implementation of the hue computation generates the result \( \text{hue}_{\text{mean}} \) as an 8-bit value as well, requiring only a sum operation and count operation for each foreground pixel. For each image, a single division is required to obtain the average. In the real-time FingerMouse system, \( \text{hue}_{\text{mean}} \) is computed for each image, and used as a reference tone in the subsequent image, assuming that this color reference will still be valid. Figure 3.20 shows the color tone sample in our sample image.

![Color Tone Sample](image)

**Figure 3.20:** *Representation of the color tone \( \text{hue}_{\text{mean}} = 15 \), measured in the sample image.*

- The *color segmentation and proximity filter component* uses the measured value \( \text{hue}_{\text{ref}} \) to perform a new foreground-classification \( S_{\text{color-combined}} \), based on the previously measured \( S_{\text{filtered}} \).
$S_{\text{color-combined}}(p)$ can be expressed as the AND-combination of two conditions described by the intermediary foreground classification functions $S_{\text{color-segmentation}}$ and $S_{\text{proximity}}$:

$$S_{\text{color-combined}}(p) = S_{\text{color-segmentation}}(p) \text{ AND } S_{\text{proximity}}(p) \quad (3.14)$$

The $S_{\text{color-segmentation}}(p)$ function expresses, whether a pixel $p$ has a color similar to the reference color tone $h_{\text{ref}}$:

$$S_{\text{color-segmentation}}(p) = 1 \iff |hue(p) - h_{\text{ref}}| < \Delta_{\text{hue}} \quad (3.15)$$

$\Delta_{\text{hue}}$ expresses the difference threshold for the color tone similarity condition, and equals $\Delta_{\text{hue}} = 10$.

The $S_{\text{proximity}}(p)$ function expresses if a pixel $p$ is in proximity of at least one horizontal neighbor distanced up to 5 pixels that has been classified to foreground by $S_{\text{filtered}}$, which corresponds to a $11 \times 1$ erosion filter. For a pixel of coordinates $(x_1, y_1)$, $S_{\text{proximity}}(x_1, y_1)$ can be expressed as a logical function, as defined in equation $3.16$.

$$S_{\text{proximity}}(x_1, y_1) = S_{\text{filtered}}(x_1 - 5, y_1)$$
$$\vdots$$
$$\text{OR } S_{\text{filtered}}(x_1, y_1)$$
$$\vdots$$
$$\text{OR } S_{\text{filtered}}(x_1 + 5, y_1) \quad (3.16)$$

(for $(x,y)$ out of the image area, we assume $S_{\text{filtered}}(x,y) = 0.$)

**Results**

Figure 3.21(c) shows the resulting output $S_{\text{color-combined}}$ (Result C). As can be seen in table 3.3, the $S_{\text{color-combined}}$ classification generates the least amount of false classifications ($N(f_p) + N(f_n)$) for the Model
Hand stereo set. It should also be noted that in the example shown in figure 3.21(c), the color segmentation generates a lot of false positives around the hand border because the background does not differentiate well from the hand hue in that area (cf. figure 3.21(b)). In most cases, less false positives will be generated.

Due to the fact that pixels classified as *foreground* by $S_{\text{filtered}}$ are likely (cf. 3.5:Results) to be true positives (belong to the hand), the computed mean hue value is accurate. The subsequent classification based on color tones works well, in cases where the foreground is colored in one consistent tone, e.g. the skin tone for a hand.

The proposed method is a color segmentation classifier in a simple form. This leads to an efficient implementation in hardware. Through the use of the $S_{\text{filtered}}$ classification to acquire the reference hue and to restrict its operation to the area close to $S_{\text{filtered}}=\text{foreground}$ pixels, it performs well, adopts automatically to changing light situations (changing the hue of the skin) and does not require an initiation phase. Color segmentation alone (without stereo vision) is widely used for hand tracking, e.g. by Imagawa et al.(1998, [13]); Kjeldsen et al.(1996, [14]); Kurata et al.(2001, [15]) and a survey by Vezhnevets et al.(2003, [50]).
Figure 3.21: Output (c) and intermediary stages (b) and (c) of the color segmentation module:
(a) segmentation results for the function $S_{\text{proximity}}$. (white = pixels classified to foreground)
(b) segmentation results for the function $S_{\text{color-segmentation}}$. (white = pixels classified to foreground)
(c) Result C: the segmentation results for $S_{\text{color-combined}}$, resulting of the processing of $S_{\text{filtered}}$ by the color segmentation module.
Coloring: true positive: white, true negative: black, false positive: yellow, false negative: red
Table 3.3: Count of false positive pixels ($N(f_p)$) and of false negative pixels ($N(f_n)$), for different segmentation functions running on the Model Hand stereo sample.

<table>
<thead>
<tr>
<th>Function</th>
<th>$N(f_n)$</th>
<th>$N(f_p)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_{CL}$</td>
<td>2261</td>
<td>2123</td>
</tr>
<tr>
<td>$S_{CR}$</td>
<td>1538</td>
<td>3224</td>
</tr>
<tr>
<td>$S_{SL}$</td>
<td>1001</td>
<td>891</td>
</tr>
<tr>
<td>$S_{SR}$</td>
<td>333</td>
<td>1763</td>
</tr>
<tr>
<td>($S_{SR} \text{ OR } S_{SL}$)</td>
<td>182</td>
<td>4705</td>
</tr>
<tr>
<td>($S_{CL} \text{ OR } S_{CR}$)</td>
<td>503</td>
<td>2902</td>
</tr>
<tr>
<td>($S_{CR} \text{ AND } S_{CL}$)</td>
<td>2683</td>
<td>809</td>
</tr>
<tr>
<td>($S_{SR} \text{ AND } S_{SL}$)</td>
<td>1127</td>
<td>490</td>
</tr>
<tr>
<td>$S_{\text{combined}}$</td>
<td>546</td>
<td>1308</td>
</tr>
<tr>
<td>$S_{\text{filtered}}$</td>
<td>1869</td>
<td>26</td>
</tr>
<tr>
<td>$S_{\text{color-combined}}$</td>
<td>204</td>
<td>850</td>
</tr>
</tbody>
</table>

3.7. Foreground Position Measurement

The foreground position measurement module computes the locations $(x_{cog,\text{filtered}}, y_{cog,\text{filtered}})$ and $(x_{cog,\text{color-combined}}, y_{cog,\text{color-combined}})$ in the right camera image, corresponding to the center-of-gravity (c.o.g.) of all the pixels classified as foreground by the $S_{\text{filtered}}$ and $S_{\text{color-combined}}$ function respectively.

$(x_{cog,\text{filtered}}, y_{cog,\text{filtered}})$ is defined as the average of all $(x,y)$-coordinate values, for all pixels $p(x,y)$, $S_{\text{filtered}}(p) = 1$:

$$
x_{cog,\text{filtered}} = \frac{1}{N_{\text{filtered}}(\text{foreground})} \cdot \sum_{\forall p, S_{\text{filtered}}(x,y)=1} x
$$

$$
y_{cog,\text{filtered}} = \frac{1}{N_{\text{filtered}}(\text{foreground})} \cdot \sum_{\forall p, S_{\text{filtered}}(x,y)=1} y \quad (3.17)
$$

$N_{\text{filtered}}(\text{foreground})$ is the number of pixels classified by $S_{\text{filtered}}$ to foreground, defined in equation 3.13. For the computation of $(x_{cog,\text{filtered}}, y_{cog,\text{filtered}})$, two values $(x,y)$ have to be summed up for each foreground pixel, as well as a pixel count, similar to the
function of the average hue computation component. These functions can be computed on-the-fly for each pixel. The division by $N_{\text{filtered}}(\text{foreground})$ is only carried out once for an entire image. It can be computed outside the FingerMouse-IC, which only needs to output the divisor (sums) and dividend ($N_{\text{filtered}}(\text{foreground})$).

The c.o.g. $(x_{\text{cog,color-combined}}, y_{\text{cog,color-combined}})$ for the output of $S_{\text{color-combined}}$ is computed in an analog fashion.

**Results**

Table 3.4 shows the measured c.o.g. coordinates $(x_{\text{cog,filtered}}, y_{\text{cog,filtered}})$ *(Result D)* and $(x_{\text{cog,color-combined}}, y_{\text{cog,color-combined}})$ *(Result E)* for the hand sample image, as well as the c.o.g. computed for the groundtruth segmentation. The same results are visualized in figure 3.22.

**Table 3.4:** The c.o.g. coordinates measured in the images (a)-(c) of figure 3.22. The images have a resolution of $320 \times 214$ pixels. The origin $(1,1)$ is situated at the upper left image corner.

<table>
<thead>
<tr>
<th></th>
<th>$x_g$</th>
<th>$y_g$</th>
</tr>
</thead>
<tbody>
<tr>
<td>ground truth</td>
<td>149</td>
<td>158</td>
</tr>
<tr>
<td><em>Result D</em>: c.o.g. for $S_{\text{filtered}}$</td>
<td>149</td>
<td>157</td>
</tr>
<tr>
<td><em>Result E</em>: c.o.g. for $S_{\text{color-combined}}$</td>
<td>150</td>
<td>153</td>
</tr>
</tbody>
</table>

**3.8. Conclusions**

- We have presented an algorithm that performs a foreground segmentation as it was specified in section 2.1. The algorithm does not need an initiation or calibration phase, so that the FingerMouse system can be used directly after powering it up. This means a higher comfort for the user, compared to system that require a user-assisted calibration procedure each time the device is used. Furthermore, such a calibration free algorithm enables to power the device only during the time it is needed, to save energy. The output quality under realistic scenario circumstances is analyzed in chapter 4.

- The presented algorithm relies on block matching methods that have been proven efficient in other real-time implementations.
3.8. Conclusions

The novelty we present is the segmentation processing layer (cf. figure 3.2, p. 21) that handles the outputs from the block matching modules: we transform the data into a single bit per pixel, and the subsequent processing does not store any disparity values. This enabled a hardware design that does not require additional buffering, and the binary segmentation data is processed by simple logic operators, as will be shown in chapter 5.
In this chapter, the output of the foreground classification and tracking algorithm from chapter 3 is evaluated using different quantitative measures.

In different simulations, the output quality is evaluated using sets of input images corresponding to the application scenarios described in section 2.2. Another simulation serves to evaluate the impact of the choice of a baseline $b$ and focal length $f$ on the system’s performance.

Since the algorithm is intended for use in a wearable or mobile system, we simulate detrimental image effects such as camera noise, motion blur and dynamic range decay, as can occur for specific lighting situations.

Another analysis simulates the impact of geometrical imprecisions that occur at the construction of an embedded stereo vision system.
4.1. System Output Quality Quantification

4.1.1. Segmentation Quality

A measure for output quality of the segmentation map is the recognition rate $R_{image}$, the ratio of correctly classified pixels. It is defined in analogy to the percentage of bad matching pixels $B$, defined in formula 3.7, p. 29 and in [42]:

$$R_{image} = \frac{N(t_p) + N(t_n)}{(X \times Y)} \quad (4.1)$$

$N(t_p)$ is the number of true positive pixels, $N(t_n)$ the number of true negative pixels in the segmentation output of an image of $X \times Y$ pixels.

In order to analyze the classification results more precisely, we define two individual recognition rates, one is calculated within the ground truth foreground area, and the other within the ground truth background.

We define the foreground recognition rate $R_{fg}$ and the background recognition rate $R_{bg}$:

$$R_{fg} = \frac{N(t_p)}{N(p, S_{gt}(p) = true)} \quad (4.2)$$

$$R_{bg} = \frac{N(t_n)}{N(p, S_{gt}(p) = false)} \quad (4.3)$$

$N(p, S_{gt}(p) = true)$ is the number of foreground pixels in the ground truth segmentation map, $N(p, S_{gt}(p) = false)$ is the number of background pixels in the ground truth segmentation map.

We define another measure, the foreground-detection signal-to-noise ratio $SNR_{fg-d}$. It expresses the ratio of the density of foreground classified pixels in the ground truth foreground area (true positives) against the density of foreground classified pixels in the background area (false positives). For an application (e.g. gesture recognition) performing post-processing of the measured foreground (the ”signal”), the $SNR_{fg-d}$ expresses, how clear this area contrasts to the detected background (the ”noise”).

$$SNR_{fg-d} = \frac{N(t_p)}{N(p, S_{gt}(p) = true)/N(p, S_{gt}(p) = false)} \quad (4.4)$$
4.2. System Evaluation for Application Scenarios

It can also be written as $SNR_{fg-d} = \frac{R_{fg}}{1-R_{bg}}$. In analogy to evaluations in stereo literature, we consider the center part of the image for the calculations and discard the left and right image border.

4.1.2. Tracking Precision

In order to quantify the precision of the centre of gravity (c.o.g.) measurement $(x_{cog}, y_{cog})$, we define the c.o.g.-deviation: the euclidean distance (in pixels) to the centre of gravity of the ground truth foreground $(x_{gt-cog}, y_{gt-cog})$:

$$\Delta_{cog} = \sqrt{(x_{cog} - x_{gt-cog})^2 + (y_{cog} - y_{gt-cog})^2}$$  

(4.5)

4.1.3. Robustness

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.

4.2. System Evaluation for Application Scenarios

4.2.1. Motivation

The motivation of the following simulation is to quantify the FingerMouse system’s performance under the following circumstances:

- sets of scenes containing different hands and different persons in the foreground, as would occur under the finger tracking and mobile video telephony background removal scenarios described in section 2.2
- the scenes are captured with an optical setup with the same field-of-view and $K$-factor\(^1\) as the FingerMouse

4.2.2. Simulation Setup

In order to test the performance of the algorithm, it is simulated in a bit-true model in Matlab, using different sets (described below) of stereo

\(^1\)\(K = (b \cdot f \cdot \frac{X_i}{x_s})\), cf. formula 3.2, p. 23. $K$ defines the disparity for a given depth $z$: $d = K \cdot \frac{1}{z}$.
images. In order to quantify the output using the measures previously introduced, manually drawn ground truth images are used.

All images are downscaled to QVGA (320 × 240) resolution for testing. The camera numbers relevant to the algorithm are shown in table 4.1.

The following sets were captured:

- **Set 1: hand**
  A human hand (different persons) is captured indoors, in different positions and with different backgrounds. This set is used as a reference in the simulations in sections 4.4, 4.5 and 4.6.

Sets 1 and 2 contain 10 stereo images, captured using a Nikon² D80 camera (gain: ISO800, aperture: f11). The camera was mounted on a sliding rail³, and left and right images were taken sequentially. The original photos were taken in full camera resolution (3872 × 2592) at a focal length $f = 18 \text{ mm}$, then subwindowed to $2954 \times 1978$ resolution. The subwindow image corresponds to the focal length $f = 23.6 \text{ mm}$, equaling the sensor width $x_s$. The resulting stereo images correspond to the field-of-view and $K$ factor of the images captured in the FingerMouse system. All images of the sets 1 and 2 are depicted in appendix A.4.1.

- **Set 2: person**
  Different persons facing the camera are captured indoors with different backgrounds.

- **Set 3: model hand at different baselines $b$, focal lengths $f$ and hand range $Z$**
  A static scene with a model hand at a range $Z = 30 \text{ cm}$ is captured indoors at different baselines, at a constant focal length $f_O = 18 \text{ mm}$. (camera: Nikon D80, gain: ISO200, exposure time: $\frac{1}{3} \text{ sec}$, aperture: f11)

  Different focal lengths $f$ are achieved by choosing a subwindow at the center of the images. With $X_{sw}$ being the width of the subwindow, $X_O$ the original image width, the new virtual focal length $f_{sw}$ equals $f_{sw} = f_O \cdot \frac{X_O}{X_{sw}}$

---

²Nikon Corporation (www.nikon.com)
³Jasper Engineering, Heavy Duty Twin Camera Slide Bar. (www.stereoscopy.com/jasper/)
Table 4.1:  *Optical parameters of the FingerMouse and of the stereo camera setup used to capture the sets*

<table>
<thead>
<tr>
<th>Camera</th>
<th>Focal length [mm]</th>
<th>Baseline [mm]</th>
<th>Sensor width [mm]</th>
<th>K factor at QVGA resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>FingerMouse</td>
<td>3.6</td>
<td>25</td>
<td>3.6</td>
<td>8</td>
</tr>
<tr>
<td>Nikon D80</td>
<td>23.6</td>
<td>25</td>
<td>23.6</td>
<td>8</td>
</tr>
</tbody>
</table>

4.2.3. Simulation Results

Table 4.2 shows the output quality measures averaged individually for sets 1 and 2 as well as the standard deviation ($\sigma$). Set 1 (pictures of hands) is evaluated with $S_{filtered}$ and $S_{color\text{-}combined}$ operation modes. Set 2 (persons) is only evaluated for $S_{filtered}$, since the color filter is not applicable to multi-colored foreground objects, such as a person.

Table 4.2: *System output evaluated for the sets 1 and 2*

<table>
<thead>
<tr>
<th>Set</th>
<th>Output</th>
<th>$R_{fg}$ [%] ($\sigma$)</th>
<th>$R_{bg}$ [%] ($\sigma$)</th>
<th>$SNR_{fg-d}$ ($\sigma$)</th>
<th>$\Delta_{cog}$ [pixels] ($\sigma$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set 1</td>
<td>$S_{filtered}$</td>
<td>75.45 (5.37)</td>
<td>99.56 (0.18)</td>
<td>212.15 (120.72)</td>
<td>4.10 (2.64)</td>
</tr>
<tr>
<td>Set 1</td>
<td>$S_{color\text{-}combined}$</td>
<td>97.71 (0.74)</td>
<td>99.71 (0.23)</td>
<td>507.89 (300.09)</td>
<td>2.48 (1.14)</td>
</tr>
<tr>
<td>Set 2</td>
<td>$S_{filtered}$</td>
<td>80.96 (2.01)</td>
<td>99.37 (0.28)</td>
<td>164.18 (101.18)</td>
<td>2.82 (2.28)</td>
</tr>
</tbody>
</table>

The numbers describe the "conservative" characteristics of our segmentation approach: pixels are only classified to foreground by $S_{filtered}$ if the confidence that the pixels belong to the foreground is high. All other pixels are classified to background. This results in a small portion (less than 1 %) of false positives: $R_{fg} > 99\%$. The foreground is detected more sparsely, $R_{fg}$ is lower than $R_{bg}$: set 1 has a $R_{fg}$ of 75\%, set 2 achieves 80 %. The differences result from the different foreground objects: the person area contain more structure than the hands.
Chapter 4: Evaluation in an Uncontrolled Environment

The foreground areas which $S_{filtered}$ does not detect, correlate to the error sources described in section 3.2.2 (p. 30): especially picture areas occupied by the hand or person showing little structure (i.e. areas of homogeneous intensity within the search window) result in false negatives. Furthermore, the noise filtering by a $11 \times 1$ erosion, in the final stage of the algorithm (cf. section 3.5), is very aggressive: besides eliminating most false positives, it also removes correct positives.

The $S_{color-combined}$ output in set 1 recognizes more of the foreground area, but also produces some more false positives, especially at the foreground object borders, as the color classification overshoots if the background near the hand has a similar color tone. The $SNR_{fg-d}$ and $\Delta_{cog}$ improve approximately by a factor 2, compared to the $S_{filtered}$ output.

Figures 4.1 and 4.2 show example images and algorithm outputs for sets 1 and 2. A complete overview of all the images and outputs in sets 1 and 2, as well as the individual recognition ratings, is presented in appendix A.4.1. The images, as well as the numbers show that the foreground recognition works very well.

In order to show the use of the produced segmentation maps we show examples of feature extraction in hand images and background removal in the person images.

When it comes to background removal, the $S_{filtered}$ segmentation map serves as a mask to mark pixel areas in $I_r$ to be blurred: the background. A first step is to find a coherent area that contains the person. It is obtained by post-processing $S_{filtered}$: we "fill up" all the background areas that are surrounded by foreground pixels. This method is always applicable, when the application is directed at a foreground area or object that is a coherent area i.e. it does not enclose areas belonging to the background.

We propose a filter generating $S_{post-processed}$ as follows: only those pixels are classified as foreground that are surrounded by at least 1 foreground pixel at a search distance $d_{fillup}$, and for at least seven out of eight search directions (up, down, left and right, as well as the diagonals). The results are presented in figure 4.3(a) ($d_{fillup} = 20$). Additionally, we apply two $3 \times 3$ dilation operations to smooth the outer border. The result, $S_{post-processed}$ is shown in figure 4.3(b). Using this segmentation map, the background in the original camera image $I_r$ can be rendered black or blurred, as shown in figure 4.4.

For the application scenario of hand tracking, the calculation of the center-of-gravity is a first feature extracted from the images that could
be used to control a pointer. However, as more of the hand becomes visible in the camera image, the c.o.g. moves to a different position of the hand. A more sophisticated approach is the tracking of a fingertip. As an example, we propose the following scheme: We look for the furthest foreground point in $S_{\text{color-combined}}$ from the c.o.g. In order to be robust against false positives, we consider only foreground pixels containing 20 or more foreground pixels within a $5 \times 5$ block around them. Figure 4.5 shows the resulting fingertip detection. Along with c.o.g. calculation, a third feature can be extracted: the orientation of the line through the c.o.g. and the fingertip. This feature can be used to determine a hand rotation in the image plane, e.g. to control pointer.
Figure 4.2: Simulation for Set 2, Pair 10
(a) and (b) are stereo input images to the simulation of the mobile video telephony background removal scenario.
(c) is the simulated segmentation map $S_{\text{filtered}}$ output by the Finger-Mouse simulation for the input image $I_l$ and $I_r$ shown in (a) and (b).
4.2. System Evaluation for Application Scenarios

Figure 4.3: Simulation of a possible scheme for post processing the foreground area computed by the FingerMouse:
(a) shows all the pixels surrounded by foreground pixels 
(b) shows $S_{post-processed}$, the result of two dilations of (a). The foreground area is now coherent and its borders are smoothed.

Figure 4.4: Simulation of images, post processed for video telephony background removal.
(a) shows the original image $I_r$, with a blackened background, while 
(b) shows a blurred background. Both methods enable to obscure the background, and the images contain less information to be encoded and transmitted through the cellular network.
The background was processed using the segmentation map shown in figure 4.3(b).
Figure 4.5: Simulation of the hand position tracking. The red cross marks the c.o.g. in $S_{\text{color-combined}}$, as computed within the FingerMouse. The green cross designates the point of the hand furthest away from the c.o.g., as measured by our simulation.

actions such as ”clicking”.

4.3. Influence of Baseline and Focal Length

4.3.1. Motivation

The FingerMouse system uses a non-variable baseline $b$ of 25 $mm$, and lenses of $f = 3.6$ $mm$ focal length.

Nevertheless, it is interesting to see how the baseline and focal length affect the performance of the algorithm. On the one hand, the FingerMouse lenses can be exchanged to lenses with other focal lengths and sub-windowing (performed by the image sensors) of the images can also create a higher virtual focal length. On the other hand, it is important to know how the system performs when scaled to smaller proportions, i.e. to a smaller baseline.

4.3.2. Simulation Setup

The simulation is performed on set 3, which contains 100 stereo images, captured at ten different baselines $b = 2$ $mm$; 4 $mm$; 6 $mm$; 8 $mm$; 10 $mm$; 15 $mm$; 20 $mm$; 25 $mm$; 45 $mm$ and 64 $mm$.

Ten different focal lengths $f$ were used: 18.6 $mm$; 23.6 $mm$; 28.6 $mm$; 33.6 $mm$; 38.6 $mm$; 43.6 $mm$; 48.6 $mm$;
4.3. Influence of Baseline and Focal Length

53.6 mm; 58.6 mm and 63.6 mm.

The hand is located 30 cm in front of the camera, and the disparity thresholds \(d_{\text{min}}\) and \(d_{\text{max}}\) are chosen such that \(Z_{\text{min}} = 20\) cm and \(Z_{\text{max}} = 40\) cm.

4.3.3. Simulation Results

The results for all combinations of base lengths are shown in figure 4.8. For some combinations the \(K\) factor is so high that no calculations were made as \(Z_{\text{prox}}\) becomes larger than the hand range of 30 cm (black squares in figure 4.8). For some other \(b\) and \(f\) combinations, the resulting \(Z_{\text{prox}}\) is between 20 cm and 30 cm, so that the near threshold \(Z_{\text{min}}\) had to be increased for their simulation (those combinations are those located next to the black squares in figure 4.8). These simulations hence contain more false negatives. All the results are presented in numbers in appendix A.4.2, p. 143. Figures 4.6 and 4.7 show \(I_r\) and the algorithm outputs for \(b = 15\) mm and \(b = 4\) mm, and for all baselines.

From \(b = 6\) mm to 45 mm, the algorithm shows good performance for all focal lengths \(f\). The slightly worse results for \(b = 64\) mm are caused by the hand range of 30 cm being too close for a setup with such a high \(K\) factor\(^4\), and parts of the hand are closer than \(Z_{\text{prox}}\). The effect is also emerging at \(b = 45\) mm.

At small baselines (< 6 mm), the system is still functional: the \(R_{bg}\) for \(S_{\text{filtered}}\) is even higher than for larger baselines. This is due to the fact that the areas containing occlusions become narrower as the baseline decreases. This results in less false positives. Therefore, \(R_{bg}\) and \(SNR_{fg-d}\) are high.

At \(b = 2\) mm and 4 mm, the simulation shows good ratings for some combinations, and less good ones for other combinations. This is a hint that the system becomes more sensitive at small baselines, as the background differentiates from the foreground only by a very small disparity.

\(^4K = b \cdot f \cdot X_i \cdot x_s^{-1}\), cf. equation 3.2, p. 23
Figure 4.6: This figure displays a part of the input images and output simulations for set 3: the input images captured at $b = 15$ mm are shown. As the focal length $f$ increases, the field of view becomes smaller. The foreground detection quality itself differs little. Each pair is shown with $I_r$ (top), $S_{filtered}$ and $S_{color-combined}$ (bottom)
4.3. Influence of Baseline and Focal Length

Figure 4.7: This figure displays a part of the input images and output simulations for set 3: the input images captured with a baseline \( b = 4 \) mm. The simulation shows that the foreground detection fails partly in some cases. Each pair is shown with \( I_r \) (top), \( S_{filtered} \) and \( S_{color-combined} \) (bottom).
Figure 4.8: Results of the simulation of different baseline and focal length combinations.

Rows, from top to bottom: $R_{fg}$; $R_{bg}$; $SNR_{fg-d}$; $\Delta_{cog}$

Columns: $S_{filtered}$ on the left, $S_{color-combined}$ on the right


4.4. Impact of Camera Alignment Tolerances

4.4.1. Motivation

In any real stereo vision system, there is a deviation from the intended ideal parallel camera geometry. The reasons for those deviations and remedies are described in section 5.2.1 (p. 99).

We intend to show and quantify how manufacturing tolerances affect the classification rate of the FingerMouse system.

Deviations can be expressed as translations and rotations along three axes, between the ideal and the actual orientation of the left camera. Figure 4.9 shows the three axes rotational errors $\Omega_{\text{tilt}-X}$, $\Omega_{\text{tilt}-Y}$ and $\Omega_{\text{rotation}}$.

![Diagram showing relative disorientation compared to an ideally parallel setup of the image sensors, expressed along three axes](image)

**Figure 4.9:** Relative disorientation compared to an ideally parallel setup of the image sensors, expressed along three axes

Translational deviations of the placement of the image sensors can also occur in three axes, most likely along the $X$ and $Y$ axis, as the PCB and image sensor ICs are located in the $X-Y$ plane.

4.4.2. Simulation Setup

In order to simulate a divergence along $\Omega_{\text{rotation}}$, the left camera image is rotated around the image center by angles from $\Omega_{\text{rotation}} = -2^\circ$ to $\Omega_{\text{rotation}} = 2^\circ$.

Translational deviations $d_x$ and $d_y$ along the $X$ and $Y$ axes are simulated by shifting the left camera images in horizontal and vertical
directions individually. At QVGA resolution, the image is translated by 1 pixel for a sensor translation distance \( d = \frac{x_s}{320} \), equaling 11.25 µm. As the pixels have a square shape, this applies for both horizontal and vertical direction. The simulation is carried out for left image sensor translations \( d \) of 0 to 136 µm in all four directions.

The two rotational deviations \( \Omega_{\text{tilt}-X} \) and \( \Omega_{\text{tilt}-Y} \) can as be approximated by a translational shift of the left camera image. At QVGA resolution, a deviation of \( \Omega_{\text{tilt}-X} \) or \( \Omega_{\text{tilt}-Y} \) by 0.1° corresponds approximately to a horizontal or vertical image translation of 0.6 pixels. The simulation covers angles from -2° to 2°, for both \( \Omega_{\text{tilt}-X} \) and \( \Omega_{\text{tilt}-Y} \).

All images are pre-processed at full resolution (2954×1978) and then downsampled to QVGA resolution for the classification simulation.

**Note**

Such an approximation of a camera rotation through a translation is only valid for small angles, as is the case in this simulation.

### 4.4.3. Simulation Results

Figure 4.10 shows the result of in plane rotation by \( \Omega_{\text{rotation}} \). \( R_{fg} \) of \( S_{\text{filtered}} \) drops almost linearly from 75% to 38% for \( \Omega_{\text{rotation}} = 2° \), as well as its \( R_{bg} \), from 99.5% to 97%. The latter points out that the number of false positives is multiplied by six, for \( \Omega_{\text{rotation}} = 2° \). The larger number of false positives falsify the hue\(_{\text{mean}}\) measurement (cf. section 3.6). This is the reason why \( S_{\text{color-combined}} \), which assures good \( R_{fg} \) up to \( \Omega_{\text{rotation}} = 0.4° \), performs worse for larger angles.

Figure 4.11 shows the result of a vertical misalignment of the image sensors. The algorithm is sensitive to this kind of misalignment: for each angular deviation of 0.17° (or a translation of \( d_y = 11.25 \, \mu m \)), the QVGA image shifts vertically by 1 line. Since the algorithm compares pixels on the same image line in \( I_r \) and \( I_l \), the chance of still finding the correct matches vanishes rapidly.

While the deviations shown in figures 4.10 and 4.11 display a more or less symmetrical behavior, the horizontal deviations \( \Omega_{\text{tilt}-X} \) and \( d_x \) shown in figure 4.12 show a very different behavior, depending on the direction of the deviation. For \( \Omega_{\text{tilt}-X} \) or \( d_x < 0 \), \( I_l \) is translated to the right. The result is an increasing disparity of all scene objects: for each shift by a 1 QVGA pixel (resulting of an angle 0.17° or a translation of \( d = 11.25 \, \mu m \)), the disparities increase by 1 pixel. In this case, the
4.4. Impact of Camera Alignment Tolerances

**Figure 4.10:** Influence of rotational deviation along $\Omega_{rotation}$

Note: In this and all following figures, the bright curves describe $S_{filtered}$, the dark curves $S_{color-combined}$.

**Figure 4.11:** Influence of rotational deviation along $\Omega_{tilt-Y}$ and of vertical translational deviation $d_y$.

foreground pixels still fall into the threshold $d_{min}$ and $d_{max}$ resulting in an unchanged $R_{fg}$ for $S_{filtered}$. Some medium distanced objects are now seen as foreground, leading to a worsening $R_{bg}$. The latter causes $S_{color-combined}$ to decrease in performance for $\Omega_{tilt-X} < -0.6^\circ$.

Deviations in the other direction ($\Omega_{tilt-X}$ or $d_x > 0$) cause the disparities to drop. Again, $S_{filtered}$ remains good for a certain range, up to $\Omega_{tilt-X} = 1^\circ$, and then degrades, as the decreased disparities do not fall
into the foreground detection range anymore. The behavior concerning $R_{bg}$ is very different: for $\Omega_{tilt-X} > 0.4^\circ$, negative disparities start to occur (an object being more to the left in $I_l$ than in $I_r$, contrary to the search direction of the block matching) and result into wrong disparity calculations. Some of those fall into the foreground range $d_{min} - d_{max}$ and cause false positives.

The $SNR_{fg-d}$ classification rating and $\Delta_{cog}$ is more symmetrical, for both $S_{filtered}$ and $S_{color-combined}$. This means the effects on $R_{fg}$ and $R_{bg}$ combine into similar $SNR_{fg-d}$ and $\Delta_{cog}$ ratings, for both deviation directions.

![Figure 4.12: Influence of rotational deviation along $\Omega_{tilt-X}$ and of horizontal translational deviation $d_x$.](image)

Image sensors like those used in the FingerMouse system, often offer the possibility to read out a rectangular subwindow of the image area. This makes it possible to countermeasure translational deviations in $X$ and $Y$ direction, as well as rotational deviations $\Omega_{tilt-X}$ and $\Omega_{tilt-Y}$. The precision of the subwindowing is determined by the sensor’s physical resolution (FingerMouse: $640 \times 480$ pixels): at the worst case, the remaining error corresponds to half the width of a sensor pixel. In the FingerMouse system, the physical pixel width equals $\frac{x_s}{640} = 5.625 \mu m$. The worst case error of half a pixel corresponds to a translational deviation of $2.8125 \mu m$ or a rotational error $\Omega_{tilt-X}$ and $\Omega_{tilt-Y}$ of $0.04^\circ$.

Using subwindowing, the effects of $\Omega_{tilt-X}$ and $d_x$ can be neglected, and the $\Omega_{tilt-Y}$ and $d_y$ deviations can only cause a smaller harm, as shown in table 4.3.
### 4.5. Sensitivity to Different Lighting Situations

#### 4.5.1. Motivation

The scenes in sets 1 and 2 were captured such that the SNR is low and without motion blur, as the scenes were static. By adjusting the exposure time, the images were captured such that their dynamic range covers the 8-bit range.

In a wearable computing or mobile application scenario, the system is exposed to a wide range of different illuminations of a scene. The light intensity $I$ that an outdoor scene with bright sunlight casts on the cameras varies over five orders of magnitude. In an APS CMOS image sensor, as are in use in the FingerMouse, the light intensity $I_{pixel}$ that hits each pixel of the image sensor is converted to a voltage $U_{pixel}$.

This conversion can be approximated as follows:

$$U_{pixel} = I_{pixel} \cdot Q \cdot T_{exposure} + U_{fpn} + U_{shot} \quad (4.6)$$

$T_{exposure}$ is the exposure time, the time during which the light of intensity $I_{pixel}$ is integrated. $Q$ is the efficiency factor. It depends on the pixel area and the quantum efficiency factor. $U_{fpn}$ is the fixed pattern noise, a voltage bias, constant for each pixel. $U_{shot}$ is the shot noise, and

---

<table>
<thead>
<tr>
<th>$\Omega_{\text{tilt-}Y ,[^\circ]} ,(d_y ,[\mu m])$</th>
<th>$R_{fg}$ [%]</th>
<th>$R_{bg}$ [%]</th>
<th>$\text{SNR}_{fg-d}$</th>
<th>$\Delta_{\text{cog}}$ [pixels]</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.036 (-2.53) $S_{\text{filtered}}$:</td>
<td>70.87</td>
<td>95.43</td>
<td>205.2</td>
<td>4.73</td>
</tr>
<tr>
<td>$S_{\text{color-combined}}$:</td>
<td>96.91</td>
<td>96.65</td>
<td>466.38</td>
<td>2.47</td>
</tr>
<tr>
<td>-0.018 (-1.26) $S_{\text{filtered}}$:</td>
<td>73.64</td>
<td>95.9</td>
<td>245.64</td>
<td>4.25</td>
</tr>
<tr>
<td>$S_{\text{color-combined}}$:</td>
<td>97.59</td>
<td>96.8</td>
<td>439.02</td>
<td>2.41</td>
</tr>
<tr>
<td>0.0 (0) $S_{\text{filtered}}$:</td>
<td>74.46</td>
<td>95.89</td>
<td>226.38</td>
<td>4.4</td>
</tr>
<tr>
<td>$S_{\text{color-combined}}$:</td>
<td>97.7</td>
<td>96.89</td>
<td>472.5</td>
<td>2.46</td>
</tr>
<tr>
<td>0.018 (1.26) $S_{\text{filtered}}$:</td>
<td>74.21</td>
<td>95.56</td>
<td>217.5</td>
<td>4.43</td>
</tr>
<tr>
<td>$S_{\text{color-combined}}$:</td>
<td>97.55</td>
<td>97.04</td>
<td>513.78</td>
<td>2.34</td>
</tr>
<tr>
<td>0.036 (2.53) $S_{\text{filtered}}$:</td>
<td>73.16</td>
<td>95.22</td>
<td>196.14</td>
<td>4.61</td>
</tr>
<tr>
<td>$S_{\text{color-combined}}$:</td>
<td>97.04</td>
<td>96.88</td>
<td>439.02</td>
<td>2.56</td>
</tr>
</tbody>
</table>

---

5active pixel sensor
Chapter 4: Evaluation in an Uncontrolled Environment

is approximately gaussian distributed. The level of $U_{\text{shot}}$ is independent for each pixel.

This voltage is amplified by a variable factor $G$ (gain) to match the input range of the 8-bit ADC:

$$U_{\text{ADC}} = G \cdot (I_{\text{pixel}} \cdot Q \cdot T_{\text{exposure}} + U_{\text{fpm}} + U_{\text{shot}}) \quad (4.7)$$

The sensitivity of the light measurement can hence be adjusted with two factors: the exposure time $T_{\text{exposure}}$ and the gain $G$.

The darker a scene is, the more one or the two factors needs to be increased. As can be seen in equation 4.7, increasing $G$ amplifies the amount of noise by the same proportion. Using higher exposure times $T_{\text{exposure}}$ does not produce more noise, but another detrimental image effect, motion blur.

The values for $G$ and $T_{\text{exposure}}$ are constrained between minimal and maximal values. For dark scenes, the camera operates with maximal values for $G$ and $T_{\text{exposure}}$, and $U_{\text{ADC}}$ does not reach the full input range of the ADC. This leads to a loss of dynamic range in the digital 8-bit images.

In bright scenes, the camera produces the best SNR and lowest motion blur, as $G$ and $T_{\text{exposure}}$ are low. However, the dynamic range of the whole scene exceeds the dynamic range of 8-bit images. As a result, objects, like the hand, may now occupy a smaller range of quantized intensity values, the contrast becomes compressed locally. The effect increases when the hand is not directly lit by the sun (in the shadow) while other objects in the background are.

4.5.2. Simulation Setup

The simulation analyzes the effects of camera noise, motion blur in vertical or horizontal direction and the compression of the dynamic range.

In order to simulate camera noise, the input images at QVGA were overlaid with gaussian noise of variance $\sigma^2$. The noise is applied individually to all pixels in $I_r$ and $I_l$, in the R, G and B color components.

The motion blur effect is simulated separately for horizontal and vertical directions of motion. The analysis simulates a rotation of the scene relatively to the camera by an angle $\Theta$. It is the product of the exposure time and angular velocity $\omega$:

$$\Theta = \omega \cdot T_{\text{exposure}} \quad (4.8)$$
In order to simulate the blur caused by this motion, the images in full resolution are overlaid with their pixel-wise translated copies, the translations covering angles up to $\Theta = 2^\circ$.

For the simulation of reduced dynamic range of the 8-bit input images, the pixel values in both input images are reduced to a fraction of the 8-bit range.

### 4.5.3. Simulation Results

Figure 4.13 and 4.14 show the effects of camera noise.

![Figure 4.13](image)

**Figure 4.13:** Influence of noise in input images. The noise is gaussian distributed with variances $\sigma^2$ from 0 to 0.004.

Figure 4.15 and 4.16 shows the effects of horizontal motion blur, figure 4.17 and 4.18 of vertical motion blur. The performance degradation is slow for $R_{fg}$, but $R_{bg}$ drops as well, causing the $SNR_{fg-d}$ of both outputs to deteriorate. The c.o.g. tracking precision remains high in all cases.

Figure 4.19 shows the recognition rates for dynamic ranges compressed to factors from 100 % (no compression) to 1% of the 8-bit range.

The algorithm loses quality when operating on contrast reduced images. Down to a range of 25%, corresponding to 6-bit images, the recognition rates decay mildly. This means, there is space for smaller dynamic ranges of the hand area, as would occur when the scene is bright, but the hand in shade.
Figure 4.14: Effect of different noise levels in the input images on the segmentation quality.
Set 1, pair is shown, with $I_r$ (top), $S_{\text{filtered}}$ and $S_{\text{color-combined}}$ (bottom)
4.5. Sensitivity to Different Lighting Situations

Figure 4.15: Influence of horizontal motion blur in input images. The camera turns right by $\Theta$ during $T_{\text{exposure}}$.

Figure 4.16: Effect of different amounts of horizontal motion blur on the segmentation quality. Set 1, pair 1 is shown, with $I_r$ (top), $S_{\text{filtered}}$ and $S_{\text{color-combined}}$ (bottom).
Figure 4.17: Influence of vertical motion blur in input images. The camera turns downwards by $\Theta$ during $T_{\text{exposure}}$.

Figure 4.18: Set 1, pair 1: different amounts of vertical motion blur each pair is shown with $I_r$ (top), $S_{\text{filtered}}$ and $S_{\text{color-combined}}$ (bottom)
4.5. Sensitivity to Different Lighting Situations

Figure 4.19: Influence of reduced dynamic range in input images. The dynamic range is expressed as the portion of the original dynamic range. \((\text{SNR}_{fg-d} \text{greater than 600 is not shown in the plot})\)

Figure 4.20: Set 1, pair 1: different amounts of dynamic range each pair is shown with \(I_r\) (top), \(S_{filtered}\) and \(S_{color-combined}\) (bottom)
4.6. Sensitivity to Radiometric Bias

4.6.1. Motivation

Within the captured images, the light is not uniformly distributed by the lenses (*vignetting*, an effect described in detail by Ray et al., 2002, [51]). Pixel vignetting is a similar effect and is caused by the light hitting the image sensor’s periphery at oblique angles (cf. Catrysse et al., 2000, [52]). Zheng et al. (2006, [53]) propose a method to determine the exact vignetting effect in an image.

For a given image region, the vignetting effect is similar in both the left and right camera image. This means, it does not necessarily affect the classification, especially if the disparity in that region is small (far away). However, since foreground objects have different positions in the left and right camera image, vignetting affects the foreground classification.

Another source of image intensity bias are radiometric differences between the left and right image sensor, leading to different intensity measurements for the same objects.

While intensity bias effects can be corrected by normalization methods, e.g. when knowing the lens characteristic, the correction of the intensity is a function of the pixel’s distance to the image sensor. In an energy efficient system however, the problem must be tackled in an efficient manner. In the FingerMouse algorithm, the OR combination with the outputs of the census module is used as a countermeasure to bias effects.

4.6.2. Simulation Setup

In order to simulate radiometric bias, we add a constant offset to all pixel values (making the pixels brighter) in the left camera image $I_l$. The offset ranges from 0 to 127. In order to process the images as 8-bit images, both $I_r$ and $I_l$ are reduced to 7-bit dynamic range (pixels values ranging from 0 to 127), before adding the offset.

The simulation is performed on the images of set 1.

4.6.3. Simulation Results

The algorithm’s recognition rates are shown in figure 4.21.

The $R_{fg}$ for $S_{filtered}$ decays rapidly, from 72% to 22% at 10% bias. $R_{fg}$ then asymptotically approaches the value $R_{fg} = 12\%$. This min-
4.6. Sensitivity to Radiometric Bias

Figure 4.21: Influence of intensity bias between left and right camera images.

The false positive pixels produced by the SAD modules deteriorate the $S_{filtered}$ $R_{bg}$ recognition rate down to 98% at 3% bias. For larger bias, the SAD modules still produce false disparity calculations, but less false positives. For a bias larger than 50%, the system output quality quantified as $SNR_{fg-d}$ is improving with increasing bias. $S_{color-combined}$ even achieves about the same $SNR_{fg-d}$ for 100% bias as for 0%, because the color filtering works well even with low $R_{fg}$ for $S_{filtered}$ as long as $R_{bg}$ is high.

The results clearly show that the system should rely only on the census based classifications when facing a large intensity bias. Otherwise, care must be taken to avoid vignetting and image sensor bias. The current algorithm can be adopted to be even more robust against bias if the input intensities are replaced by the gradient of the intensities along the horizontal images lines.
In this chapter we show the implementation of our approach into an embedded stereo-vision system that performs at high image processing rates while achieving lower power consumption, size and latency than other existing stereo-camera systems.

The focus lies on the architecture of an IC we specifically designed for the system: we show how its tightly coupled data buffering and parallel processing units enables the execution of the algorithm with minimal effort in terms of buffering and power consumption.
5.1. Architecture of the FingerMouse-IC

5.1.1. Overview of IC architecture

We present the key elements to the design of the FingerMouse-IC, structured in 3 layers:

- **image input layer**
  This layer contains the stereo camera interface and stores image data for on-the-fly processing\(^1\) by the subsequent layers. It is designed to store the minimum amount of data necessary for the algorithm execution.

- **depth mapping layer**
  This layer computes disparity measurements, as described in section 3.2. It is designed to operate in real-time on-the-fly using data from the image input buffers.

- **segmentation processing layer**
  This layer transforms the measured disparities into binary foreground / background classifications and generates two output segmentation maps and foreground position measurements. It performs the algorithm steps described in sections 3.4.1 to 3.7.

The IC is designed to operate at a clock frequency\(^2\) \(f_{IC} = 80\) MHz. It was designed in 2005 in collaboration with the Integrated Systems Laboratory (IIS) at ETH Zürich, as a semester thesis ("FingerMouse 6") project, involving three students and two more graduate researchers.

5.1.2. Image Input Layer

**Synchronous Camera Interface**

Two cameras in the system deliver a continuous image output stream directly to the ASIC. The images are transmitted line by line, starting at the top left pixel, over a parallel interface, as 8 bits per camera clock cycle for each camera.

The ASIC outputs the operating clock \(f_{camera}\) for the cameras:

---

\(^1\)By *on-the-fly processing* we mean the real-time processing of a continuous image data stream.

\(^2\)We denote \(\tau_X\) the clock cycle time of denoted frequencies \(f_X\), e.g. \(\tau_{IC}\) for the IC’s clock cycle time.
5.1. Architecture of the FingerMouse-IC

\[ f_{\text{camera}} = \frac{1}{8} \cdot f_{IC} \] (5.1)

This results in the pixel stream being synchronous to the processing in the ASIC. Synchronization signals are used to recognize image, row and pixel start times within the pixel stream, making the interface independent of image sensor timing properties, such as the blanking interval time between image rows (horizontal blanking interval or horizontal retrace).

The interface can operate in two modes:

- the black&white mode processes 8-bit greyscale images (8 bit per pixel)
- the color mode additionally processes color information, coded in YUV 4:2:2 format (16 bit per pixel, two camera clock cycles transmission time)

The FingerMouse-IC architecture is able to process images at a maximum pixel rate \( f_{\text{pixel,max}} \):

\[ f_{\text{pixel,max}} = \frac{1}{16} \cdot f_{IC} \] (5.2)

This pixel rate equals also the maximum pixel output rate of a color camera clocked at \( f_{\text{camera}} \), with a pixel transmission time of two camera clock cycles, as is the case in the implemented FingerMouse system. Pixel rates \( f_{\text{pixel}} \) lower than \( f_{\text{pixel,max}} \) can be processed as well.

Optional Image Downsampling

The internal image resolution \((X_i, Y_i)\) is limited horizontally to \(X_{i,max} = 340\) pixels. A downsampling module converts higher camera resolutions \((X_{cam}, Y_{cam})\) by reducing the horizontal resolution (line width) by a factor 1:1 (no reduction), 2:1, or 4:1.

The maximum camera resolution is 1360×1020 (assuming 4:3 format), and is internally processed in 340×1020 resolution. Any resolution up to that can be configured to the ASIC as well as different image aspect ratios.
Color De-Multiplexing

The pixel data is coming in as two \( YUV\ 4:2:2 \) streams from the image sensors. The stream is demultiplexed into intensity values (8 bit per pixel) and color values (16 bit U,V for two neighboring pixels).

The 16 bit color information is converted into a single 8-bit value \( hue(p) \), describing the color tone of the pixel \( p \) as well as of its right neighbor pixel. The FingerMouse architecture hence stores color information at half the horizontal resolution, compared to the intensity information. The color information from the left camera is discarded, since only hue values from the right camera image are used for the color processing (cf. section 3.6).

The remaining amount of color information (4 bit per pixel) equals one fourth of the original color information transmitted by the camera (8 bit per pixel for left image and for right image), thus reducing the amount of buffering and IC circuit area needed.
5.1. Architecture of the FingerMouse-IC

Input Pixel Buffers

The internal intensity row buffer stores the intensity information of four complete horizontal stereo lines (rows) in a FIFO buffer: while the row \( y = n \) is read into the buffer, the three previous rows \( y = (n - 3), y = (n - 2) \) and \( y = (n - 1) \) remain completely buffered. The data contained in these three rows is exactly the data needed (reference and candidate windows) by the depth mapping layer to calculate the disparities of all the pixels of the row \( y = (n - 2) \) in both the left and the right camera image.

The buffer stores up to \( X_{i,\text{max}} = 340 \text{ stereo-pixels}^3 \) per row and is organized in four discrete on-chip RAM sections, each storing one row of stereo-pixels. Each section is sized 688 bytes and has a 64-bit word interface. The three active stereo row RAMs can deliver 12 stereo-pixels (4 stereo triplets, 192 bits) at each IC clock cycle, equaling a throughput of \( TP_{IRB} = f_{IC} \cdot 192 \text{ bits} \).

In a similar fashion, a color row buffer stores the hue values for four rows.

Image Input Layer: Results

- the synchronous camera interface does not require any buffering to synchronize left and right camera data

- along with the downsampling module, the image input layer offers a high flexibility in the choice of camera resolutions and frame rates.

- the color de-multiplexing reduces the incoming image data to 62.5 % (100% of the intensity information and 25 % of the color information) of the raw camera data, by discarding information not needed in the subsequent layers

- the described pixel input buffers, containing three complete image rows, represent the smallest amount of buffering necessary to compute block matching with a block height and a search windows height of \( (2H + 1) = 3 \text{ pixels} \) for pixels of the middle row of the three rows. While a system using a frame-buffer\(^4\) for the two stereo-pixel, we mean two pixels of the same coordinate in the two corresponding stereo images. By (stereo-)triplet, we mean three vertically neighboring (same x coordinate) (stereo-)pixels.

\(^4\)A frame-buffer is a buffer containing the data of a complete image.
complete QVGA\(^5\) stereo images would need 307 kilobytes of RAM, the FingerMouse image input buffer size is 3440 bytes.

5.1.3. Depth Mapping Layer

In order to process the incoming pixel data on-the-fly, the four disparity functions \(D_{SAD,R \gg L}, D_{census,R \gg L}, D_{SAD,L \gg R} \) and \(D_{census,L \gg R},\) defined in equations 3.4 and 3.5, p. 26) have to be computed for one pixel within the duration \(\tau_{pixel,max} = 16 \times \tau_{IC}.\)

During those 16 \(\tau_{IC}\) cycles, the four correlation functions are computed for all values \(t = 0..47\) (\(d_{\text{limit}} = 47\)). Each \(C(t)\) computation compares the reference block to a \(3 \times 5\) pixel candidate block. In order to read the five stereo-triplets of the candidate block from the intensity row buffer, two clock cycles are necessary. Assuming the correlation results would be obtained directly after the data is read, it would still take \(2 \times (d_{\text{limit}} + 2W + 1) = 104\) pixels (= twice the width of the complete search window) clock cycles to read the necessary data from the RAM, too long for on-the-fly operation.

Intermediary Cache and Parallelization of Block Matching

In order to achieve the necessary disparity computation speed \((4 \times f_{pixel,max}\) disparity computations per second) at the target IC operating frequency \(f_{IC},\) a higher degree of parallelization along with an intermediary register buffer is used. For each correlation function \(C(t),\) four parallel block comparison units execute a block comparison at each clock cycle, delivering the results \(C(t), C(t+1), C(t+2)\) and \(C(t+3).\)

The data for those 16 block comparison units is delivered from two special buffers, the search block ring buffers (one for \(R \gg L\) and one for \(L \gg R\) matching), consisting of registers. Their organization as a ring buffer takes advantage of the fact that candidate blocks of neighboring pixels overlap. The ring buffers contain three sections, each holding four triplets. At each clock cycle, four new stereo-triplets are read from the intensity row buffer (this equals the SRAM’s maximum throughput \(TP_{IRB}\)) into one section of both buffers, while the other two sections contain eight neighboring stereo-triplets. The eight buffered triplets contain four overlapping candidate blocks (cf. figure 5.2) and both search block ring buffers are directly connected to eight block comparison units for \(R \gg L\) matching and eight for \(L \gg R\) matching.

\(^5\)QVGA commonly denotes a resolution of \(320 \times 240\) pixels.
5.1. Architecture of the FingerMouse-IC

Two more buffers, the $R \gg L$ and $L \gg R$ reference block buffers store the reference window ($5 \times 3$ pixels) needed for each block comparison.

While the disparity measurements $D_{SAD,R\gg L}(p_1)$ and $D_{census,R\gg L}(p_1)$ for a pixel $p_1(x_1, y_1)$ in the right camera image $I_R$ are computed according to figure 5.2, the $L \gg R$ units measure the disparity of a pixel $p_2$ located in the left camera image $I_L$ at $(x_2, y_2) = (x_1 + d_{\text{limit}}, y_1)$. $p_1$ and $p_2$ share the same search window coordinates in the left and right camera image respectively: this guarantees the necessary execution speed, as both search block ring buffers are filled with the same stereo-triplets. However, the results from the $R \gg L$ and $L \gg R$ cannot be directly fused. This fusion is described in section 5.1.4.

After the 16 clock cycles have elapsed, 48 block comparisons for each correlation method have been computed, and the best matches determine the results $D_{SAD,R\gg L}(p_1)$, $D_{census,R\gg L}(p_1)$, $D_{SAD,L\gg R}(p_2)$ and $D_{census,L\gg R}(p_2)$.
Chapter 5: Embedded Implementation

Figure 5.3: Depth mapping layer schematic
(in blue: buffers, in green: processing units)

During the following 16 \( \tau_{IC} \) clock cycles, disparities are calculated for two new candidate pixels: the right neighbors (x coordinate incremented by 1) of \( p_1 \) and \( p_2 \). In this fashion, the circuit computes four disparity results at each \( \tau_{pixel} = 16 \cdot \tau_{IC} \), leading to the Results A, represented in figure 3.6 (p. 28).

Depth Mapping Layer: Results

- The number of 48 (= \( d_{limit} + 1 \)) block comparisons is a result of the fact that the search block ring buffers contains four blocks during only 12 of the 16 cycles, since the buffers have to be filled at the beginning. The choice of 16 \( \tau_{IC} \) clock cycles per disparity computation equals double the camera clock period \( \tau_{camera} = 8 \cdot \tau_{IC} \). These clock cycle relations guarantee the absence of overhead that would occur if \( \tau_{pixel} \) was not a multiple of \( \tau_{camera} \).

- The implemented depth mapping layer processes \( 48 \times 4 \times f_{pixel, max} \)
= 960 \cdot 10^6 \text{ block comparisons/s.}

- Each SAD block comparison unit performs 15 subtractions and 15 accumulate operations (cf. formula 3.4, p. 26) and processes 15 bytes of data during one IC clock cycle; each census block comparison unit performs 24 pixel intensity comparisons, 24 binary XNOR operations and 24 accumulate operations (cf. formula 3.5, p. 27) and processes 15 bytes of data.

To achieve 960 \cdot 10^6 \text{ block comparisons/s}, a sequential processing architecture (e.g. a standard CPU) would need to execute at least \(48.96 \cdot 10^9 \text{ operations/sec}\), and load \(14.4 \cdot 10^9 \text{ bytes/s}\) from RAM.

### 5.1.4. Segmentation Processing Layer

#### Left-Right Fusion

As described in the previous section, the depth mapping layer produces SAD and census based disparity measurements for a pixel \((x_0, y_0)\) in \(I_R\) and for a pixel \((x_0 + d_{\text{limit}}, y_0)\) in \(I_L\).

Using a thresholding function \(S\) (introduced in section 3.3, p. 32), the binary segmentation results \(S_{SR}(x_0, y_0)\) and \(S_{CR}(x_0, y_0)\) are computed, and fused to

\[
S_R(x_0, y_0) = S_{SR}(x_0, y_0) \text{ OR } S_{CR}(x_0, y_0)
\]  

Concurrently, \(S_L(x_0 + d_{\text{limit}}, y_0)\) is computed as:

\[
S_L(x_0 + d_{\text{limit}}, y_0) = S_{SL}(x_0 + d_{\text{limit}}, y_0) \text{ OR } S_{CL}(x_0 + d_{\text{limit}}, y_0)
\]

In order to fuse \(S_R\) and \(S_L\), a FIFO buffer is used, the left-right fusion buffer, consisting of \(d_{\text{limit}} + 1 = 48\) bits in registers (cf. figure 5.4).

If, and only if, the result \(S_L\) equals true \((\text{foreground})\), it is stored in that buffer, at the index \(i_{\text{in}} = D_{\text{census}, L \gg R}(x_0 + d_{\text{limit}}, y_0)\). After each pixel, the buffer is shifted to the right. The result is that the buffer contains \(S_L(x_0, y_0)\) at index \(i_{\text{out}} = 47\).

\(^6\)The actual number is higher, due to overhead such as loop jump operations.

\(^7\)The choice for \(D_{\text{census}, L \gg R}\) over \(D_{\text{SAD}, L \gg R}\) was motivated by superior classification ratings in a simulation carried out before the IC was manufactured.
Figure 5.4: Segmentation processing layer schematic  
(in blue: buffers, in green: processing units)

on the 48 preceding $L \gg R$ disparity measurements, for pixels between $(x_0, y_0)$ and $(x_0 + d_{\text{limit}}, y_0)$.

We now obtain $S_{\text{combined}}(x_0, y_0)$, the segmentation resulting of the four correlation units:

$$S_{\text{combined}}(x_0, y_0) = S_R(x_0, y_0) \text{ AND } S_L(x_0, y_0) \quad (5.5)$$
5.1. Architecture of the FingerMouse-IC

Noise Filtering

This component corresponds to the algorithms presented in section 3.5, p. 41.

The $S_{combined}$ bits are stored in a noise filter buffer $REG_{noise-filter}$, a 5-bit FIFO register buffer. The AND-operation of the 5 bits leads the filtered result $S_{filtered}$ (cf. equation 3.11, p. 42).

This result is one of the segmentation map outputs of the IC, Result B (cf. figure 3.17, p. 45). It is stored in a 16-bit register $REG_{result-filtered}$. This enables 16-bit wide IC data output at a frequency $f_{SM-out} = \frac{1}{16} \cdot f_{pixel}$ (cf. section 5.1.5), and access for the color segmentation fusion module (cf. next section).

Optional Color Segmentation Fusion

This component corresponds to the algorithms presented in section 3.6, p. 45.

- At each second pixel cycle $\tau_{pixel}$, a 8-bit $hue(p)$ value is read from the color row buffer, valid for two pixels.

- During each $\tau_{pixel}$, the average hue computation component accumulates the 8-bit hue values for a pixel $p$, for which $S_{filtered}(p) = true$ into a 25-bit register $REG_{hue-sum}$ and counts those pixels in a 17-bit register $REG_{n-foreground-filtered}$.

After a full image has been processed, during the vertical blank interval time, the average hue $hue_{mean}$ is obtained by a division of $REG_{hue-sum}$ by $REG_{n-foreground-filtered}$ and stored in a 8-bit register $REG_{hue-mean}$ (cf. formula 3.12, p. 47).

- During each $\tau_{pixel}$, $hue(p)$ is compared to $REG_{hue-mean}$ and the proximity function is derived by AND operations of bits in the $REG_{result-filtered}$ register. If the hue difference is below the threshold $\Delta_{hue} = 10$ and at least one of all 10 neighboring pixels has been classified to true (foreground), $S_{color-combined}$ is set to true.

This result is the second of the two segmentation map outputs of the IC, Result C (cf. figure 3.21, p. 50). It is stored in a 16-bit register $REG_{result-color-combined}$. 
Foreground Position Measurement Component

This component corresponds to the algorithms presented in section 3.7, p. 51.

- During each $\tau_{\text{pixel}}$, the c.o.g. computation component accumulates the 9-bit values $x$ and $y$ for each pixel $p(x, y)$, for which $S_{\text{filtered}}(p) = \text{true}$ into 26-bit registers $\text{REG}_{x-\text{sum}-\text{filtered}}$ and $\text{REG}_{y-\text{sum}-\text{filtered}}$. The foreground pixels ($S_{\text{filtered}}(p)=1$) are counted in a 17-bit register $\text{REG}_{n-\text{foreground}-\text{filtered}}$.

- During each $\tau_{\text{pixel}}$, the c.o.g. computation component accumulates the 9-bit hue values $x$ and $y$ for each pixel $p(x, y)$, for which $S_{\text{color-combined}}(p) = \text{true}$ into 26-bit registers $\text{REG}_{x-\text{sum}-\text{color-combined}}$ and $\text{REG}_{y-\text{sum}-\text{color-combined}}$. The foreground pixels ($S_{\text{color-combined}}(p)=1$) are counted in a 17-bit register $\text{REG}_{n-\text{foreground}-\text{color-combined}}$.

- After a full image has been processed, during the vertical blank interval time, the center-of-gravity coordinates are calculated by four divisions and are stored in four 11 bit registers $\text{REG}_{\text{cog}-x-\text{filtered}}$, $\text{REG}_{\text{cog}-y-\text{filtered}}$, $\text{REG}_{\text{cog}-x-\text{color-combined}}$, $\text{REG}_{\text{cog}-y-\text{color-combined}}$. These values represent Results D and F, shown in figure 3.22 (p. 53) and have a resolution of 11 bits (quarter subpixels of the internally processed image). The four values are output via a RS232 interface.

Segmentation Processing Layer: Results

The segmentation processing layer does not buffer any disparity values. All data is directly converted into binary segmentation bits and the proposed fusion method, the filtering and the c.o.g. computation are designed such that the computations are done on-the-fly by logical operations, running at $f_{\text{pixel, max}}$. The total amount of bits buffered in registers equals 317 (realized in flip-flops).

The FingerMouse architecture we described shows how the restriction to post-processing in the binary foreground / background segmentation domain allows a design with much less circuitry (both in processing and buffering), compared to systems carrying out depth map fusion and post-processing of disparities. This is reflected by the power consumption of the IC, being lower than the power consumed by the other stereo vision systems (presented at the end of this chapter).
5.1.5. Interfaces

- segmentation and depth map output
  This interface outputs either $S_{\text{filtered}}$ or $S_{\text{color-combined}}$ over a 16-bit wide interface. Each 16-bit word contains classifications of 16 pixels. The words are transmitted at a rate $f_{SM-out}$:
  $$f_{SM-out} = \frac{1}{16} \cdot f_{\text{pixel,max}} = \frac{1}{256} \cdot f_{IC} = 0.3125 \text{ words/s}.$$  
  Three additional signals carry synchronization data for pixel, line and frame synchronization, identical to the camera synchronization signals. The sync signals and the low (compared to the pixel rate) word-rate $f_{SM-out}$ ease interfacing to embedded CPUs.
  Alternatively, the interface can also output either $D_{SAD,R \gg L}$ with $D_{\text{census},R \gg L}$ or $D_{SAD,L \gg R}$ with $D_{\text{census},L \gg R}$. Two disparity measurements for one pixel are transmitted per word, with a word rate $f_{DM-out} = f_{\text{pixel,max}}$.

- configuration and tracking output interface
  A bi-directional RS-232 interface transmits, for each frame, four 11-bit values, the coordinates of the center-of-gravity. The interface is also used to set algorithm parameters, such as the internal resolution $(X_i,Y_i)$, the depth thresholds $d_{\text{min}},d_{\text{max}}$ and the choice whether to use color information.

5.1.6. FingerMouse-IC Prototype Specifications

The FingerMouse-IC was designed with the *Synopsis Design Compiler*, using standard cell libraries by *Virtual Silicon*\(^8\). The design was simulated with *Mentor Modelsim* and the routing of the circuit was done using *CADENCE Silicon Ensemble*.

The IC was manufactured by UMC\(^9\) in 2005. Table 5.1 provides an overview of its key specifications, table 5.1.6 shows the fabrication process data and figure 5.5 shows the ASIC layout. Further tests, the operation with different core voltages $V_{DD}$ and different IC clock frequencies $f_{IC}$ is presented in appendix A.5 (p. 146).

---

\(^8\) *Virtual Silicon* has been bought up by *Mosaid* meanwhile.

\(^9\) UMC (United Microelectronics Corporation) was founded as Taiwan’s first semiconductor company in 1980. UMC is the second biggest manufacturer of integrated circuits wafers for fabless semiconductor companies. ([http://www.umc.com.tw/](http://www.umc.com.tw/))
### Table 5.1: *FingerMouse IC Specifications*

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>supply voltage</td>
<td>$2.5 , V ; 3.3 , V$ (core voltage $V_{DD}$; I/O voltage)</td>
</tr>
<tr>
<td>core size</td>
<td>$1887 \mu m \times 1887 \mu m$ (including global and power routing)</td>
</tr>
<tr>
<td>chip technology</td>
<td>umcL250, 250 nm</td>
</tr>
<tr>
<td>pin count</td>
<td>28 input; 22 output; 20 power; 14 empty</td>
</tr>
<tr>
<td>on-chip RAM</td>
<td>32 Kbits (28 KBits used)</td>
</tr>
<tr>
<td>transistor count (without RAM)</td>
<td>approx. 380 000</td>
</tr>
<tr>
<td>image data rate</td>
<td>$5 , M\text{pixel/s at } f_{IC} = 80 , MHz$ or $6.25 , M\text{pixel/s at } f_{IC} = 100 , MHz$</td>
</tr>
<tr>
<td>power dissipation at full</td>
<td>$78 , mW$ at $f_{IC} = 80 , MHz$ or $96 , mW$ at $f_{IC} = 100 , MHz$</td>
</tr>
<tr>
<td>processing rate</td>
<td></td>
</tr>
</tbody>
</table>

### Table 5.2: *FingerMouse IC Fabrication Process Specifications*

<table>
<thead>
<tr>
<th>Process Step</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manufacturer</td>
<td>UMC</td>
</tr>
<tr>
<td>Technology node</td>
<td>250 nm</td>
</tr>
<tr>
<td>Technology name</td>
<td>L250</td>
</tr>
<tr>
<td>Substrate Well Formation</td>
<td>twin wells on $p^{-}$-sub, optional Epitaxial Wafer</td>
</tr>
<tr>
<td>Device isolation</td>
<td>Shallow Trench Isolation</td>
</tr>
<tr>
<td>Gate oxide thicknesses</td>
<td>$5 , nm, 6.5 , nm$ ($3.3 , V$)</td>
</tr>
<tr>
<td>Silicon Rsh reduction</td>
<td>titanium salicided</td>
</tr>
<tr>
<td>Interconnect layers</td>
<td>1P/5M</td>
</tr>
<tr>
<td>ILD Planarization</td>
<td>Chemical Mechanical Polishing</td>
</tr>
<tr>
<td>Interconnect material</td>
<td>W(plug), AICu</td>
</tr>
<tr>
<td>Poly Pitch</td>
<td>600 nm</td>
</tr>
<tr>
<td>Metal Pitch</td>
<td>M1 = 640 nm, M2-M4 = 800 nm, MT = 960 nm</td>
</tr>
<tr>
<td>Core voltage</td>
<td>$2.5 , V$</td>
</tr>
<tr>
<td>IO voltage</td>
<td>$3.3 , V$</td>
</tr>
<tr>
<td>Leakage</td>
<td>$3 , pA/\mu m$(n-ch), $2 , pA/\mu m$(p-ch)</td>
</tr>
</tbody>
</table>
5.2. Architecture of the Embedded FingerMouse System

5.2.1. Optical Setup

Operation Range and Field of View

For a FingerMouse system design using a specific image sensor, the camera baseline $b$ and the focal length $f$ of the cameras determine the operation range ($Z_{\text{prox}}$ to $+\infty$) and the field of view ($\alpha_x, \alpha_y$) of the system.

The operation range of the system is defined by $Z_{\text{prox}}$, the minimal distance for which objects can still be perceived by the FingerMouse:

$$Z_{\text{prox}} = K \cdot \frac{1}{d_{\text{limit}}} \, [m], \quad \text{where } K = b \cdot f \cdot \frac{X_i}{x_s}$$

The FingerMouse, operates with the following values:

$d_{\text{limit}} = 47$, $b = 25 \, mm$, $x_s = 3.6 \, mm$ (image sensor width), $f = 3.6 \, mm$.

$X_i$ corresponds to the internal horizontal processing resolution. At a resolution of e.g. $X_i = 320 \, pixels$, $Z_{\text{prox}}$ equals $17 \, cm$ ($K=8$).
The resolution of the disparities calculated in the FingerMouse system is defined by the disparity range (0 to 47 pixels) and the quantization step of 1 pixel. There are 48 depth values corresponding to the 48 disparity values, according to equation 3.1 (p. 23). From that equation, it follows that the quantization steps between measured ranges $Z$ increase with increasing range. The depth resolution, equaling the inverse of the quantization step, is highest the closer the range is to $Z_{prox}$, as seen in figure 5.6(b).

Figure 5.6 also reveals how the depth measuring range and resolution is dependent on the choice of the factor $K$ ($K = 8$ in the FingerMouse system).

\[ \alpha_x = 2 \cdot \arctan \left( \frac{x_s}{2 \cdot f} \right) \quad , \quad \alpha_y = 2 \cdot \arctan \left( \frac{0.75 \cdot x_s}{2 \cdot f} \right) \] (5.6)

The field of view of the of a single camera equals:

\[ \alpha_x = 2 \cdot \arctan \left( \frac{x_s}{2 \cdot f} \right) \quad , \quad \alpha_y = 2 \cdot \arctan \left( \frac{0.75 \cdot x_s}{2 \cdot f} \right) \] (5.6)
FingerMouse. Hence, the field-of-view of either of the two cameras of the FingerMouse equals:

\[ \alpha_x = 53.13^\circ, \quad \alpha_y = 41.11^\circ \]

The field of view of the FingerMouse system is equal to the overlapping portion of both the left and right camera’s field of view. As can be seen in figure 5.7, the horizontal field of view of the FingerMouse has the same angular extent \( \alpha_x \) as a single camera, its position is translated to the right by \( \frac{b}{2} \), and forward (along the \( Z \)-axis) by \( b \). The forward translation by \( b \) is only the case for the FingerMouse focal length \( f = x_s \).

![Figure 5.7: Field of view of the FingerMouse system, seen from top. The horizontal field of view of the FingerMouse is shown in blue.](image)

**Precision of the Parallel Stereo Camera Setup**

While the algorithm presented in chapter 3 is based on a camera model with two perfectly parallel cameras (cf. appendix A.3), any real stereo camera systems shows some divergence from the ideal setup. This divergence can be expressed as three angles \( \Omega_{rotation} \), \( \Omega_{tilt-X} \) and \( \Omega_{tilt-Y} \) as shown in figure 5.8.
The impact of these deviations was analyzed in section 4.4: $\Omega_{\text{rotation}}$ and $\Omega_{\text{tilt}-Y}$ have an important influence on the functioning of the algorithm.

The divergence in a real system can have different causes:

- PCB placement: due to tolerances in the manufacturing process, the placement of the image sensors can result in divergence
- packaging: the image sensor die may not be parallel to the package, again due to production tolerances
- optics: a different relative placement of the optics of the two cameras can affect the image alignment

Several solutions exist to ensure proper alignment:

- In prototype development, alignment can be adjusted mechanically: the cameras can be mounted on a flexible boards, that can be adjusted via the screws holding it until the output images show no divergence.
- Tilt deviations $\Omega_{\text{tilt}-X}$, $\Omega_{\text{tilt}-Y}$ can be corrected by an image translation. In order to achieve such a translation without additional buffering, the windowing function of the image sensors

Figure 5.8: Relative disorientation compared to an ideally parallel setup of the image sensors, expressed along three axes (this is a repetition of figure 4.9)
can be used: it enables the output of a sub-window of the complete sensor area. Two sub-windows of the same size in the respective cameras can be used with the desired translation. New sensors, like the Micron MT9P031 ([54]), integrate electronic pan and tilt functions that enable subwindowing at a higher resolution than the downsampled output image, allowing even smaller corrections.

- If the two previous solutions are not applicable, a rectification module is needed that performs the necessary adjustment. As a pre-processor, such a module is concatenated in the system between the cameras and the processing unit. A rectification calculates the re-projection of the tilted image such that $\Omega_{rotation} = 0$, $\Omega_{tilt-X} = 0$ and $\Omega_{tilt-Y} = 0$ for the re-projection. Ayache et al. (1988, [55]) and Loop et al. (1999, [56]) present methods for rectifying stereo images.

### 5.2.2. Embedded FingerMouse System

#### Layout of the Embedded System

Figure 5.9 shows a basic system layout required to operate the FingerMouse ASIC. A single clock generation circuit provides the operating clock signal to the ASIC, which then provides the camera clock.

A micro-controller (a TI MSP430F1611 is used in the FingerMouse) performs the following tasks:

- it configures the ASIC (image resolution, color mode, etc.) via RS232

- it configures the cameras (operation mode, exposure control schemes, etc...) and synchronizes them via a reset command. It can also throttle the frame rate (pixel transmission speed) by setting clock pre-scaler functions in the image sensors (if available).

- it serves as an interface to a host computer, receiving external parameters and transmitting tracking results, e.g. via RS232

In this design, the segmented images are transmitted as a raw digital signal. Another possibility is to route this data through the micro-controller, requiring a faster interface than RS232. The proposed system layout can also be a part of a larger system (e.g. a mobile phone), where
the FingerMouse delivers segmentation information as an addition to a processing unit that already does image processing.

![Diagram of FingerMouse Embedded System Layout](image)

**Figure 5.9:** *FingerMouse embedded system layout: components and interfaces*

**System Boot Up Time**

When powered, the FingerMouse system starts working within a fraction of a second. The image sensors are the components determining the startup time: during the duration of a few frame transmissions, the exposure time is regulated to match to dynamic range of the scene to be captured. This regulation process and its duration is dependent on many factors, such as lighting conditions and image resolution. During this process, the system outputs results, but correct results occur only once the cameras reach their normal operation. This happens in less than a second, e.g. after 5 frames, equaling a delay of 166 ms at a frame rate of 30 frames/s.

The short boot-up time enables the system to be powered up only when needed for user interaction. A boot-up delay in the range of less than 200 ms is practically not noticeable by the user.

**Architecture of the Embedded FingerMouse System: Results**

The described embedded architecture was implemented in a prototype system, the IC-FingerMouse prototype.

A micro-controller (TI MSP430 F1611) controls the cameras (Omnivision OV7649, low voltage color CMOS VGA imager, lenses: Sekonix
board lenses, \( f = 3.6 \, mm, b = 25 \, mm, x_s = 3.6 \, mm \), configures the FingerMouse IC and transmits the tracking results via a RS232 interface. The segmented images are output continuously by the IC over a parallel interface. The system includes power regulation from a battery voltage input.

To optimize the size of the system, it was implemented on a four-layer PCB, using chip-scale package image sensors and with FingerMouse IC die directly bonded onto the PCB.

The size of the resulting PCB is \( 43 \, mm \times 18 \, mm \) (cf. figure 5.10(a)), its thickness is \( 8 \, mm \) (at the lenses). The power consumption of the complete system is \( 166 \, mW \). \( (2 \times 30 \, mW \) camera power, \( 5 \, mW \) MSP power, \( 23 \, mW \) for clock generation, \( 78 \, mW \) for the FingerMouse IC) With a power regulation efficiency of \( 89\% \), it draws \( 187 \, mW \) from the batteries.

An evaluation prototype, sized \( 15 \, cm \times 11 \, cm \), was implemented with the same components, partly in larger chip packaging, and including testing interfaces (cf. figure 5.10(b)). The system was tested successfully for functionality, by comparing the transmitted camera images with the system outputs in a running system, verifying that the output matches the simulated results.

The results related to the prototype were published in [57], [58] and [59].

![Figure 5.10:](a) FingerMouse small scale prototype PCB (next to 2 Euro coin) (b) evaluation prototype
5.3. Other Real-Time Stereo Systems and Comparison

5.3.1. Earlier FingerMouse Prototypes (DSP, FPGA based)

Prior to the FingerMouse system presented in this thesis, two other prototypes were implemented, the *DSP-FingerMouse* ([60]) and the *FPGA-FingerMouse*. They run a simplified stereo vision algorithm, designed only at retrieving a center-of-gravity measurement of the foreground. Since the two systems are larger and consume more power than the IC based FingerMouse, while the c.o.g. measurement is worse, the two older systems are less efficient than the IC based FingerMouse.

The architecture of the DSP-FingerMouse and the FPGA-FingerMouse are described in the appendix A.7, and the architecture details are compared to the FingerMouse IC architecture.

5.3.2. Embedded Stereo Systems

Research Prototypes

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

1. The "Miniature Stereo Vision Machine (MSVM-III)" by Jia et al.(2004, [61]), runs stereo-vision depth mapping at 120 frames per second at 320 × 240 pixel resolution (twice the pixel rate of the FingerMouse). It uses three image sensors, and an FPGA for the vision processing. It is shown in figure 5.11 (a).

2. Darabiha et al.(2003, [62]) have implemented a real time stereo vision system running on four Xilinx Virtex2000 FPGAs. It is based on a phase-based stereo vision algorithm. The used algorithm is described by Fleet et al.(1994, [63]). The system produces depth maps at a rate of 30 frames per second, a resolution of 256 × 360 pixels, and 256 disparity levels. The high number of disparity levels is used to measure disparities at sub-pixel precision. It is shown in figure 5.11 (b) and (c).

3. An implementation on a hardware of several Xilinx 4013 FPGAs is described by Corke et al.(1999, [64]). It achieves 30 256×256 frames per second, of a maximal disparity 32.
5.3. Other Real-Time Stereo Systems and Comparison

4. The "PARTS Reconfigurable Computer" (Woodfill et al., [65], 1997) is a general-purpose reconfigurable machine with wide I/O memory, consisting of 16 Xilinx 4025 FPGAs 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. The system computes 15M block comparisons per second, with 22 W power consumption.

5. The "Stereo Machine for Video-rate Dense Depth Mapping" (Kanade et al., [66], 1996) performs 30M block comparisons per second, (FingerMouse: 960 M block comparisons per second) e.g. processing 200×200 images at 30 fps and a disparity range of 32. It features image rectification and can use two to six input cameras. The system uses an array of TMS320C40 DSP processors.

More embedded solutions using DSPs and/or FPGAs are described by Porter et al.(1997, [67]) and Ayache et al.(1991, [68]).

State-of-the-Art Commercial Stereo Vision Modules

Today, commercial embedded systems are available:

1. **Stereo on a Chip** (STOC) is an embedded implementation of a real-time stereo engine on an FPGA board. The algorithms have been developed by Konolidge et al. at the *SRI International Artificial Intelligence Lab*\(^\text{10}\) (already publishing on a DSP based embedded prototype in 1997, [69]). The hardware is commercialized

---

\(^{10}\)www.ai.sri.com
by Videre Design\textsuperscript{11} as a complete stereo smart camera (cf. figure 5.12), transmitting both standard images (left camera) and disparity maps. The system was released to the market in December 2005.

![Image](a) The Stereo on a Chip processing board
(b) The complete stereo camera system (the processing board is integrated inside), dimensions: 132 mm $\times$ 44 mm $\times$ 40 mm (image courtesy: SRI)

The system can process up to 30 frames/s at $640 \times 480$, with a disparity search width of 64 pixels, and with correlation block sizes of $15 \times 15$. The processing unit is a Xilinx Spartan 3 - 1000 with 512 KB of SRAM, clocked at $88 \text{ MHz}$. At this rate, the system consumes 2.4 W of power (0.9 W are consumed by the processing unit, the FPGA). \textsuperscript{[70]}

2. PC/104+ nDepth Vision System by Focus Robotics\textsuperscript{12}.

This PC/104 form factor system computes stereo vision from two externally connected cameras.

3. DeepSea G2 Vision System by TYZX\textsuperscript{13}.

This is a large scale ($3.8 \text{ cm} \times 28.3 \text{ cm} \times 16.9 \text{ cm}$, 22 cm baseline), high performance stereo vision system. It captures 200 frames/s, at a resolution of $2048 \times 512$ pixels. Its power consumption is 14 W, which allows for automotive applications or integration into large robots, but is too high for wearable computing.

\textsuperscript{11}www.videredesign.com
\textsuperscript{12}www.focusrobotics.com
\textsuperscript{13}www.tyzx.com
5.3.3. PC/Software Based Stereo Systems

Since a few years, real-time implementations are achieved on standard PCs. This has become possible through the increase of the PC’s performance, the use of SIMD (single instruction, multiple data) processor technologies like MMX and SSE (on x86 platforms), better cache architectures and multi-core processors. SIMD optimized real-time stereo vision algorithms are presented by Hirschmüller et al. (2002, [71]) and Mühlmann et al. (2002, [72]).

The main performance increase results from algorithm optimizations: since coherent objects in a scene are coherently projected onto the image sensor, 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 hierarchical steps, using a priori information from the previous step to decrease the amount of necessary block comparisons and thus increase speed. Anandan et al. present such a hierarchical framework (1989, [73]). More recent publications include those of Van Meerbergen et al. (2002, [74]) and Sun et al. (2002, [75]).

Many different software implementations exist, e.g.:

1. **Triclops SDK** is a commercial software by the company **Point Grey Research**. It does real-time stereo block matching in high quality on a PC and can be used with **Bumblebee**, a stereo camera head by the same company. These pre-calibrated products enable fast prototyping of depth computing applications. According to the manufacturer’s website (May 1st, 2006), the software achieves a frame rate of 31 fps, at 320×240 resolution and a disparity depth of 48, running on a 2.4 GHz Pentium IV.

2. **E-Stereo** is an example of a freely available implementation, a ”C++ library for real-time disparity estimation. The library contains various functions for dense stereo matching from 2 or 3 rectified images and 3D scene reconstruction.”, by David Demirdjian ([76], [77]). On a 1.7 GHz Pentium IV, with 320×240 input images, a disparity depth of 32, it processes 14-18 frames per second, according to the author.

3. GPU based stereo vision. Some projects use the processing power of a PC’s graphical processor to run stereo block matching algorithms, e.g. in [78] (2003) by Yang et al., a **NVIDIA GeForce4**

---

14 www.ptgrey.com
graphics card in a PC achieves 50-70 million block comparisons per second. (FingerMouse: 960 million)

5.3.4. Comparison of the FingerMouse- and Other Real-Time Stereo Vision Systems

This section compares the performance, latency and power efficiency of similar systems using different architectures. All those systems are capable of performing a foreground / background segmentation. Since those systems perform the same task with slightly different methods and output quality, the figures only give an approximate comparison.

The following systems are compared:

- IC based FingerMouse (at 80MHz and at 100MHz ASIC clock)
- DSP based FingerMouse
- FPGA based FingerMouse
- PC based implementation of the FingerMouse stereo block matching algorithm
  
  We implemented a software version of the FingerMouse depth mapping layer, described in section 3.2, on a Pentium PC. The code was written in C++ and optimized for the processor’s cache architecture. On a Pentium M (@1.6GHz), the processing of 1 VGA image takes >3 seconds. (The algorithm was[79])

- PC based implementation of performance optimized stereo block matching
  
  For our comparison, we chose E-stereo as an example of an optimized PC based implementation. E-stereo is a C++ implementation (by David Demirdjian, cf. [76]) of a stereo block matching algorithm, and an additional disparity map filtering pass (not included in the FingerMouse).

- The Stereo on a Chip system (presented in section 5.3.2) as an example of a state-of-the art commercial embedded stereo system.

- The DeepSea G2 system (presented in section 5.3.2) as an example of a system with a high pixel processing speed.
• Additionally, we show numbers for the *Swissranger SR-3000 system*, designed at CSEM. ([80], [81]) This system is not a stereo vision system, but a *time-of-flight range camera*. Using this principle, depth maps are acquired, which can also be used for foreground / background segmentation (cf. appendix A.8). A disadvantage is that it operates indoors only. It measures depth of 176 \times 144 pixels at a frame rate of 25 frames/s. Range is measured from 0.3 - 7.5 m, at a precision of 7.5 cm. The principle of time-of-flight imaging is described in [21], [22].

For the PC-based system, we assume that a 2007 power efficient PC processor, e.g. a Pentium M ULV\textsuperscript{15} or Core processor, would still have a TDP\textsuperscript{16} larger than 5 W.

Table 5.3 shows an overview of the performances.

**Results**

The numbers in table 5.3 reveal that the FingerMouse system outperforms its competitors in terms of size, power consumption, processing efficiency and latency. When it comes to energy efficiency (energy per pixel), the FingerMouse is more than four times more efficient than the high speed *DeepSea G2* system and more than six times more efficient than the *STOC* system working at a similar pixel rate.

In figure 5.13, the FingerMouse system is compared to two state-of-the art competitors: the *STOC* as the least power consuming commercial stereo system, and the *DeepSea G2* system, which has the best efficiency (energy per pixel) of the commercial devices.

Figure 5.13 shows that the FingerMouse is scaled down by more than an order of magnitude in terms of system size and power consumption, while maintaining a pixel rate in the same order as the *STOC*.

\textsuperscript{15}ULV stands for ultra low voltage. This is the least power consuming Intel Pentium processor available.

\textsuperscript{16}Thermal Design Power - the power dissipation as specified by the manufacturer, applies to normal operating conditions.
Table 5.3: Comparison between the different stereo vision platforms

<table>
<thead>
<tr>
<th>Platform</th>
<th>CPU; power (CPU)</th>
<th>$f_{\text{pixel}}$ [Mpixels per second]</th>
<th>power efficiency ($W / \text{pix.}$) [$nJ$]</th>
<th>system size ($mm^3$)</th>
<th>latency (camera to output)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSP based FingerMouse</td>
<td>DSP (TI TMS320 VC33); 130 mW</td>
<td>0.5</td>
<td>260</td>
<td>$73 \times 64 \times 35$</td>
<td>13.5 ms (+65 ms for image transm.)</td>
</tr>
<tr>
<td>FPGA based FingerMouse</td>
<td>Xilinx Spartan II; 1 W</td>
<td>10</td>
<td>100</td>
<td>$69 \times 49 \times 45$</td>
<td>&lt;1 ms</td>
</tr>
<tr>
<td>PC (brute)</td>
<td>Pentium; &gt;5W</td>
<td>0.2</td>
<td>&gt;2500</td>
<td>N/A</td>
<td>(&gt;50 ms)</td>
</tr>
<tr>
<td>PC (optimized)</td>
<td>Pentium; &gt;5W</td>
<td>2.2-2.8</td>
<td>&gt;1750</td>
<td>N/A</td>
<td>(&gt;50 ms)</td>
</tr>
<tr>
<td>Stereo on a Chip system</td>
<td>Xilinx spartan III 1000; 0.9 W</td>
<td>9.2</td>
<td>97.6</td>
<td>$132 \times 44 \times 40$</td>
<td>unknown</td>
</tr>
<tr>
<td>DeepSea G2 system</td>
<td>unknown; 14 W</td>
<td>200</td>
<td>70</td>
<td>$283 \times 169 \times 38$</td>
<td>unknown</td>
</tr>
<tr>
<td>Swissranger SR-3000 system</td>
<td>t.o.f.-image sensor system; 12 W</td>
<td>0.63</td>
<td>19000</td>
<td>$67 \times 50 \times 42$</td>
<td>unknown</td>
</tr>
<tr>
<td>IC based FingerMouse</td>
<td>FingerMouse IC (80MHz); 78 mW</td>
<td>5</td>
<td>15.6</td>
<td>$43 \times 18 \times 8$</td>
<td>&lt;1 ms</td>
</tr>
<tr>
<td>IC based FingerMouse</td>
<td>FingerMouse IC (100MHz); 96 mW</td>
<td>6.25</td>
<td>15.3</td>
<td>$43 \times 18 \times 8$</td>
<td>&lt;1 ms</td>
</tr>
</tbody>
</table>
5.3. Other Real-Time Stereo Systems and Comparison

Figure 5.13: System comparison chart comparing the FingerMouse to two state-of-the-art commercial systems
In this outlook, we present and discuss the achievements of our design, and we show how the FingerMouse architecture will evolve as semiconductor manufacturing evolves. Decreasing size and power consumption of ICs are beneficial to the wearable system. Progress in CMOS image sensor manufacturing leads to higher imaging and segmentation quality. Furthermore we discuss the scalability and miniaturization prospects of the FingerMouse system.
6.1. Current Achievement

The stand-alone FingerMouse system presented in this thesis (section 5.2.2, results) qualifies as a wearable HCI system thanks to its small size, low power consumption and good usability as an instantly usable system. The instant use is possible since the system does not require user calibration before use and because of the unnoticeable boot-up time.

If the system is to be powered by its own battery, the 187 mW power consumption allows operation over hours with a small battery: with state-of-the-art lithium batteries offering an energy density of up to 3.6 kJ per cm$^3$ (Brodd et al., 2004, [82]), a battery of 1 cm$^3$ volume can power the system for 5.3 hours of continuous system operation. Since the system can be powered down when no user interaction takes place, the practical operation time is higher, or a smaller battery can be used.

6.1.1. Use as a Subsystem in a Mobile Device

Besides its use as a stand-alone wearable HCI system, the FingerMouse architecture can be embedded as a subsystem in any mobile device that makes use of foreground segmentation. The size and power consumption of the FingerMouse components qualify it for use in size and power consumption critical devices, such as mobile phones, PDAs or wearable computers. Since the FingerMouse subsystem also offers the standard image outputs of its cameras, existing vision subsystems (present in most mobile devices) can be enhanced by adding foreground detection.

The current design of the FingerMouse-IC can be extended by adding more RAM, such that the complete color information of at least one camera is buffered over three image lines. This would allow to output standard camera images and the segmentation results as a synchronous stream.

6.2. Scalability of the FingerMouse Architecture

If the performance (pixel rate, disparity range) of the current FingerMouse architecture is lower than is required by a specific application, it can be scaled to increase the resolution in five dimensions: vertical resolution ($Y_i$), horizontal resolution ($X_i$), disparity range ($[0..d_{\text{limit}}]$) and temporal resolution (frame rate $f_{\text{frame}}$).
The scaling is achieved adopting either a higher operating frequency $f_{IC}$ or a larger circuit area $A_{IC}$, both options causing a higher power consumption $P_{IC}$.

**Scaling by $f_{IC}$**

Increasing the frequency $f_{IC}$ by a factor $k$ allows one of the following:

- increase of the frame rate $f_{frame}$ by a factor $k$
- increase of the vertical resolution $Y_i$ by a factor $k$
- increase of the horizontal resolution $X_i$ by a factor $k$, up to $X_{i,max} = 340$ pixels
- a combination of the $f_{frame}$, $X_i$, $Y_i$ increase, e.g. a 2-D resolution increase (both $X_i$ and $Y_i$ are scaled by a factor $k$) by a factor of $\sqrt{k}$

Increasing the operating frequency $f_{IC}$ increases the minimal core voltage $V_{DD}$ required to operate the IC for a given manufacturing technology, as shown in appendix A.5 (p. 146). This leads to an increase of the power consumption $P_{IC}$ by a factor proportional to $(V_{DD})^2$.

**Scaling by $A_{IC}$**

Increasing the circuit area $A_{IC}$ by a factor $k$ allows one of the following:

- increase of the frame rate $f_{frame}$ linear with $k$
- increase of the vertical resolution $Y_i$ linear with $k$
- increase of the horizontal resolution $X_i$ linear with $k$
- increase of the disparity range $([0..d_{limit}])$ linear with $k$ to $([0..k \cdot d_{limit}])$
- a combination of the $f_{frame}$, $X_i$, $Y_i$, $d_{limit}$ increase, e.g. a 3-D resolution increase ($X_i$, $Y_i$ and $d_{limit}$ are scaled by a factor $k$, resulting in an unchanged near operating range $Z_{prox}$) linear with $\sqrt[3]{k}$

The power consumption $P_{IC}$ increases by the factor $k$. The calculations are described in the appendix A.6.
Conclusion

Scaling by $A_{IC}$ is more power efficient than scaling by $f_{IC}$, as the parallelization is more efficient than a clock frequency increase.

6.3. Impact of Semiconductor Evolution on the FingerMouse Architecture

6.3.1. Device Scaling and Power Consumption Reduction

CMOS manufacturing technology is constantly evolving to produce circuits of a higher density, the so-called device scaling. This has been described by Moore’s Law (Moore et al., 1965, [83]), who stated in the 1960s that the capacity of DRAMs quadruples approximately every three years (which Moore reviews in 1995, [84]). The scaling is possible by decreasing geometric dimensions and voltage levels such that electric fields are maintained constant (Dennard et al., 1972, [85]). For a given circuit, the power consumption benefits from the downscaling.

Although the classical device scaling meets physical limitations, it is currently ongoing, thanks to newer production methods and materials (cf. Horowitz et al., 2005, [86]). The recent evolution and a prediction of future industrial CMOS technology is portrayed in the *International Technology Roadmap for Semiconductors (ITRS)*, published every two years by a global consortium (2007 ITRS, [87]). Some of the ITRS numbers are presented in table A.6, making it possible to predict the size of the FingerMouse IC for a given technology. The FingerMouse IC uses 380,000 transistors (2.23 $mm^2$ at L250) plus 32768 SRAM cells (4 KByte, 1.23 $mm^2$ at L250, 6 transistors per cell).

An important factor for power consumption of newer IC generations is the decreasing supply voltage $V_{DD}$ used in low-power IC design. This leads to less energy being dissipated.

Using the ITRS trend numbers, it is possible to predict the area $A_{IC}$ and power consumption $P_{IC}$ of the FingerMouse IC using state-of-the-art manufacturing, for a given year. In figure 6.1, we show the calculated $A_{IC}$ and $P_{IC}$ for the years 1999 to 2022. The $A_{IC}$ calculations are based on the FingerMouse IC transistor and SRAM cell count, and the ITRS numbers for the transistor density of routed logic and SRAM circuits in ”MPU cost-performance product generations”. The $P_{IC}$ calculations are based on the power measurements of the L250 FingerMouse IC at $V_{DD} = 2.5 \, V \, (P_{IC} = 78 \, mW)$ and the ITRS projections of core voltages used in low operating power ICs. We assume power scaling for
both the logic circuit and the SRAM cells to be proportional to \((V_{DD})^2\). The numbers and formulas used to produce figure 6.1 are presented in appendix A.9.

\[\text{power consumption} \quad \text{size (area) of} \]
\[
\begin{array}{ll}
\text{FingerMouse IC} & \text{FingerMouse IC} \\
[mW] & [mm^2]
\end{array}
\]

![Graph showing size and power projections of the FingerMouse IC according to the ITRS trend numbers](image)

**Figure 6.1:** Size and power projections of the FingerMouse IC according to the ITRS trend numbers
*Note: we neglect the different scaling properties of IC peripherals, such as interface pads and connections*

### Conclusions

Today’s manufacturing technology allows the realization of the FingerMouse processing circuit in a 65 nm process, at a size \(A_{IC}\) of 0.3 \(mm^2\), a supply voltage \(V_{DD}\) of 0.8 V and a power consumption \(P_{IC}\) lower than 10 mW (cf. figure 6.1).

In the near future, these figures will continue to drop according to the ITRS roadmap predictions. The trend of decreasing power requirements is especially beneficiary to the mobile usability of the FingerMouse.

The gains in size and power leave room to scale the architecture to the application’s need in performance, by using more transistors and higher operating frequencies \(f_{IC}\).
6.3.2. Evolution of CMOS Image Sensors

The development of CMOS image sensors is subject to constant improvements comparable to the general advances in CMOS manufacturing technology.

The trend shows that newer sensors achieve lower power consumption $P_{\text{sensor}}$, increasing spatial and temporal resolution ($X_i, Y_i, f_{\text{frame}}$) and/or a better output quality (higher $SNR$, dynamic range and sensitivity).

Sensors with very high frame rates have been developed and described by Furuta et al. (2006, [88]), Krymski et al. (1999, [89]), Stevanovic et al. (2000, [90]) and Kleinfelder et al. (2001, [91]). These sensors transmit, with a power consumption as low as 50 $mW$, up to 1 $G\text{pixel/s}$, more than can be handled by the FingerMouse IC without scaling it. However, at lower image resolutions, high frame rates offer the possibility of high frame rate foreground detection. The several megapixels of spatial resolution $X \times Y$ in today’s state-of-the-art image sensors exceeds the needs of HCI applications, which is oriented towards the resolution of the mobile display (up to 1 $M\text{pixel}$). Nevertheless, the increased resolution is beneficial to a FingerMouse system, since the CMOS sensors also offer the possibility to read out a smaller area-of-interest image and/or to downsample the image before outputting it.

As a wearable or mobile system, the FingerMouse can encounter scenes exceeding the dynamic range of its sensors (a standard CMOS image sensor offers a dynamic range of 40-60 $dB$). Several methods have been proposed and implemented to increase the dynamic range and $SNR$, such as well-capacity adjusting (Decker et al., 1998, [92]), multiple capture (Yang et al., 1999, [93]), time-to-saturation (Stoppa et al., 2002, [94]) and self-reset (McIlrath et al., 2001, [95]). High dynamic range sensors allow the FingerMouse to recognize areas of high illumination, otherwise over-saturated and not containing structure. In case of sensors transmitting the image in a finer quantization than the 8-bit used in the the FingerMouse, a local contrast or dynamic range reduction (Tumblin et al. propose a method in [96], 1999) compresses the dynamic range, while preserving local contrast. A comparison of the different dynamic range increasing methods and the resulting $SNR$ figures are presented in the surveys by Yang et al. (1999, [97]) and Kavusi et al. (2004, [98]).

Improvements of the $SNR$ and the sensitivity of CMOS image sen-
sors are crucial to allow vision systems such as the FingerMouse to operate in low-light environments. Techniques for low noise CMOS image sensing are described by Kawai et al. (2004, [99]) and Das et al. (2004, [100]). Highly sensitive image sensors have been shown by Croft et al. (1999, [101]) and are analyzed by Degerli et al. (2000, [102]) and Goy et al. (2001, [103]).

Device scaling and process technology evolution also lead to a trend of higher integration of CMOS image sensor circuits, which also reduces the size of the optics needed to capture images. However, new challenges arise as downsizing of the image sensitive areas in the pixel elements leads to less light being captured. Those challenges are described by Lule et al. (2000, [104]) and Wong et al. (1996, [105]). The downsizing effects have to be balanced by improving the sensitivity of the sensor elements, leading to the same efforts than those performed for SNR improvement. For those reasons, image sensors are currently manufactured in processes using larger structures than those for other ICs, e.g. 180 nm (Hsu et al., 2005, [106]; Wuu et al., 2001, [107]).

Conclusion

Today’s high speed image sensors offer the possibility to scale the FingerMouse architecture to the image resolution and frame rate required by HCI applications.

Tomorrow’s image sensors will enable the FingerMouse architecture to be operational in a more and more illumination-independent fashion, as sensitivity and dynamic range improves. At the same time, the sensors will dissipate less power, and their size including optics shrinks thanks to device scaling and more sensitive sensing elements.

6.3.3. System Integration and Packaging

While the current prototype has been realized on a standard four layer PCB, further size reductions can be achieved by integrating the system in an MCM or SiP device.

However, a more drastic size reduction can be achieved thanks to the fact that the image sensors are manufactured in a CMOS process, just as the system’s processing components.

This allows the integration of processing circuits with image sensors into a single integrated circuit (called computational sensor) using a monolithic process (a study by Chen et al., 2001, [108], reviews such processes). E.g., the sensors in the FingerMouse system use an
on-chip circuit to transform the color output from \textit{bayer pattern raw} to \textit{YUV 4:2:2}. Researchers have implemented image sensors with integrated vision processing, such as \textit{optical motion flow estimation} (Mehta et al., 2006, [109]; Arreguit et al., 1996, [110]; Ancona, 1996, [111]), image compression (Nishikawa et al., 2007, [112]; Kawahito et al., 1999, [113])), DRAM frame buffer integration (Bidermann et al., 2003, [114]) and programmable units for pixel operations (Paillet et al., 1998, [115]; Ruedi et al., 2003 [116]). Ni et al. (2000, [117]) developed a smart sensor for line-based stereovision applications (such as the FingerMouse), offering different image pre-processing filters.

The FingerMouse architecture can be implemented as a single-circuit \textit{SoC} device containing the two image sensors, the current IC circuit, the micro-controller and potentially more circuitry. The IC can be put in a package containing two adequate lenses on top. An even higher degree of integration of the optics is possible: Denyer et al. (1993, [118]) show the bonding of miniature lenses directly on the silicon surface of the IC. Packages with lenses are described in the patents [119] and [120].

The SoC integration offers additional benefits: the parallel orientation of the image sensor areas within the circuits will be much more precise than in a process using two separate image sensor ICs. Radiometric differences between the two imagers will be much smaller, compared to two individual cameras manufactured separately and using individual power supplies.

While the size of a FingerMouse SoC could be scaled down according to the industry trends, an inherent property of the FingerMouse limits the smallest possible form factor: the two image sensors still need a baseline separation. We have shown in section 4.3 that hand segmentation, based on the FingerMouse algorithm, is possible with a baseline $b$ as small as 4\,mm.

\textbf{Conclusion}

The minimum width of a future FingerMouse system is determined by the baseline $b$, the diameter of the optics and the packaging process. Figure 6.2 shows the footprint of a virtual system with a 4\,mm-baseline and 1\,$\times\,$1\,mm$^2$ image sensing areas. A SoC of this size can be realized in 180\,nm technology.
Figure 6.2: Footprint of a theoretical 4 mm-baseline FingerMouse SoC. (a) shows the layout in actual size, (b) shows an image enlarged by a factor of ten.
Appendix
A.1. Curriculum Vitae

Personal Information

Patrick de la Hamette  
Born 7 November 1976, Luxembourg, Luxembourg  
Citizen of the Grand Duchy of Luxembourg

Education

2001-2008  
PhD studies in information technology and electrical engineering at ETH Zurich, Switzerland
2001-2001  
Diploma studies in electrical engineering at ETH Zurich, Switzerland
1989-1996  
Secondary school, Lycée de Garçons de Luxembourg, Luxembourg
1983-1989  
Primary school, Walferdange, Luxembourg

Work Experience

2001-2008  
Research and teaching assistant at Electronics Laboratory, ETH Zurich, Switzerland
2001-2007  
Employee at annual job fair Polymesse, ETH Zurich, Switzerland
2001  
Assistant at the Chair of Technology and Innovation Management, ETH Zurich, Switzerland
1997-2000  
Teaching assistant at Power Electronic Systems Laboratory, ETH Zurich, Switzerland
1999  
Industrial internship with SES ASTRA S.A., satellite operator, Luxembourg
1996  
Industrial internship with CFL, train network operator, Luxembourg
### A.2. Glossary

#### A.2.1. Abbreviations

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC</td>
<td>analog-to-digital converter</td>
</tr>
<tr>
<td>APS</td>
<td>active pixel sensor</td>
</tr>
<tr>
<td>ASIC</td>
<td>application-specific integrated circuit</td>
</tr>
<tr>
<td>CMOS</td>
<td>complementary metal-oxide-semiconductor</td>
</tr>
<tr>
<td>c.o.g.</td>
<td>center-of-gravity</td>
</tr>
<tr>
<td>DRAM</td>
<td>dynamic random access memory</td>
</tr>
<tr>
<td>DSP</td>
<td>digital signal processor</td>
</tr>
<tr>
<td>FIFO</td>
<td>first in, first out</td>
</tr>
<tr>
<td>FPGA</td>
<td>field-programmable gate array</td>
</tr>
<tr>
<td>GUI</td>
<td>graphical user interface</td>
</tr>
<tr>
<td>HCI</td>
<td>human computer interaction</td>
</tr>
<tr>
<td>HMD</td>
<td>head-mounted display</td>
</tr>
<tr>
<td>IC</td>
<td>integrated circuit</td>
</tr>
<tr>
<td>ITU-T</td>
<td>International Telecommunication Union, telecommunication standardization sector</td>
</tr>
<tr>
<td>MCM</td>
<td>multi chip module</td>
</tr>
<tr>
<td>PCB</td>
<td>printed circuit board</td>
</tr>
<tr>
<td>PDA</td>
<td>personal digital assistant</td>
</tr>
<tr>
<td>PSNR</td>
<td>peak signal-to-noise ratio</td>
</tr>
<tr>
<td>QBIC</td>
<td>QBIC belt integrated computer ([7])</td>
</tr>
<tr>
<td>QVGA</td>
<td>quarter VGA (popular term for 320 × 240 pixel resolution)</td>
</tr>
<tr>
<td>RMS</td>
<td>root mean square</td>
</tr>
<tr>
<td>SAD</td>
<td>sum of absolute differences</td>
</tr>
<tr>
<td>SIMD</td>
<td>single instruction, multiple data</td>
</tr>
<tr>
<td>SiP</td>
<td>system in package</td>
</tr>
<tr>
<td>SNR</td>
<td>signal-to-noise ratio</td>
</tr>
<tr>
<td>SoC</td>
<td>system-on-a-chip</td>
</tr>
<tr>
<td>SRAM</td>
<td>static random access memory</td>
</tr>
<tr>
<td>SSE</td>
<td>streaming SIMD extensions</td>
</tr>
<tr>
<td>TDP</td>
<td>thermal design power</td>
</tr>
<tr>
<td>t.o.f.</td>
<td>time of flight</td>
</tr>
<tr>
<td>ULV</td>
<td>ultra low voltage</td>
</tr>
<tr>
<td>VGA</td>
<td>video graphics array</td>
</tr>
</tbody>
</table>
A.2.2. Functions and Variables

\( A_{IC} \)  
size (area) of integrated circuit (p. 114)

\( b \)  
camera baseline (p. 23)

\( B \)  
percentage of bad matching pixels (p. 29)

\( C(t) \)  
correlation function of correspondence search (p. 25)

\( C_{census}(t) \)  
census based correlation function (p. 27)

\( C_{SAD}(t) \)  
SAD based correlation function (p. 26)

\( d \)  
disparity (p. 21)

\( d_{fillup} \)  
segmentation map fill-up search distance (p. 63)

\( d_{limit} \)  
disparity search width (p. 25)

\( d_{max} \)  
upper foreground disparity threshold (p. 32)

\( d_{min} \)  
lower foreground disparity threshold (p. 32)

\( d_x \)  
horizontal displacement of image sensor (p. 69)

\( d_y \)  
vertical displacement of image sensor (p. 69)

\( f \)  
focal length (p. 23)

\( f_{camera} \)  
clock frequency of the camera ICs (p. 85)

\( f_{frame} \)  
frame rate (p. 114)

\( f_{IC} \)  
clock frequency of the FingerMouse integrated circuit (p. 84)

\( f_{pixel} \)  
pixel rate (p. 85)

\( f_{pixel, \max} \)  
maximum pixel rate (p. 85)

\( f_{SM-out} \)  
IC segmentation data output rate (p. 93)

\( G \)  
gain for pixel voltage in APS sensor (p. 74)

\( H \)  
half-height of of reference window (p. 25)

\( hue_{mean} \)  
mean hue of foreground in right camera image (p. 47)

\( I \)  
light intensity (p. 73)

\( i_{in} \)  
left-right fusion buffer input index (p. 91)

\( i_{out} \)  
left-right fusion buffer output index (p. 91)

\( I_l \)  
image captured by the left camera (p. 24)

\( I_r \)  
image captured by the right camera (p. 24)

\( I_{pixel} \)  
light intensity hitting pixel in APS sensor (p. 73)

\( k \)  
scaling factor (p. 114)

\( K \)  
ratio between the disparity \( d \) and the depth \( z \), (p. 23)

\( P_{IC} \)  
power consumption of integrated circuit (p. 114)
A.2. Glossary

$P_{shift}$ perspective shift transformation (p. 36)

$Q$ efficiency factor (p. 73)

$R$ RMS error, between disparity calculations and the ground truth disparities (p. 29)

$R_{bg}$ background recognition rate (p. 56)

$R_{fg}$ foreground recognition rate (p. 56)

$R_{image}$ ratio of correctly classified pixels (recognition rate) (p. 56)

$S$ segmentation function (p. 32)

$S_L$, $S_R$, $S_{SR}$, $S_{CR}$, $S_{SL}$, $S_{CL}$ intermediary segmentation maps (p. 37)

$S_{color-combined}$ segmentation map output of algorithm, using additional color segmentation (p. 48)

$S_{filtered}$ segmentation map output of algorithm (p. 41)

$SNR_{fg-d}$ foreground-detection signal-to-noise ratio (p. 56)

$t$ offset in correspondence search (p. 25)

$t_0$ offset minimizing correspondence function (p. 25)

$T_{exposure}$ exposure time of the image capture (p. 73)

$TP_{IRB}$ maximum throughput of intensity row buffer SRAM (p. 88)

$U_{ADC}$ amplified pixel voltage in APS sensor (p. 73)

$U_{fpn}$ fixed pattern noise voltage in APS sensor (p. 73)

$U_{pixel}$ pixel voltage in APS sensor (p. 73)

$U_{shot}$ shot noise voltage in APS sensor (p. 73)

$V_{DD}$ IC core voltage (p. 96)

$W$ half-width of of reference window (p. 25)

$X, Y, Z$ axes of the real word coordinates (p. 20)

$X_{cam}, Y_{cam}$ camera resolution (p. 85)

$x_{cog}, y_{cog}$ coordinates in the image of the foreground center of gravity (p. 11, 51)

$X_i, Y_i$ internal image processing resolution (p. 85)

$X_{i, max}$ maximum internal horizontal image processing resolution (p. 85)

$x_s, y_s$ physical dimensions of the sensitive area of the image sensor (p. 98)

$z$ depth (or range) (p. 23)
\begin{tabular}{ll}
\textbf{$Z_{max}$} & upper foreground depth threshold (p. 10 , p. 32) \\
\textbf{$Z_{min}$} & lower foreground depth threshold (p. 10 , p. 32) \\
\textbf{$Z_{prox}$} & minimal operating range (p. 10 , p. 97) \\
\textbf{$\alpha_x$} & horizontal field-of-view (p. 98) \\
\textbf{$\alpha_y$} & vertical field-of-view (p. 98) \\
\textbf{$\Delta_{cog}$} & deviation of center-of-gravity measurement (p. 57) \\
\textbf{$\Delta_{Hamming}$} & hamming distance (p. 27) \\
\textbf{$\Delta_{hue}$} & difference threshold for color tone similarity (p. 48) \\
\textbf{$\Theta$} & angle of rotation of the system during exposure time (p. 74) \\
\textbf{$\sigma$} & standard deviation (p. 59) \\
\textbf{$\sigma^2$} & variance (p. 59) \\
\textbf{$\tau_{IC}$} & clock period of the FingerMouse integrated circuit (p. 84) \\
\textbf{$\tau_{pixel}$} & processing time of one pixel (p. 85) \\
\textbf{$\Phi$} & census transform (p. 27) \\
\textbf{$\omega$} & angular velocity (p. 74) \\
\textbf{$\Omega_{tilt-X}$} & horizontal tilt angle of left camera (p. 69, p. 99) \\
\textbf{$\Omega_{tilt-Y}$} & vertical tilt angle of left camera (p. 69, p. 99) \\
\textbf{$\Omega_{rotation}$} & rotation angle of left camera along its optical axis (p. 69, p. 99) \\
\end{tabular}
A.2.3. Definitions

active pixel sensor (APS) image sensor consisting of an integrated circuit containing an array of pixel sensors, each pixel containing a photodetector and an active amplifier.

center-of-gravity the center-of-gravity of a set of pixels is defined as the average of their positions.

computational sensor image sensor with processing circuits in a single IC.

dilation, erosion dilation and erosion are two fundamental operations in morphological image processing from which all other morphological operations are based.

depth map a pixel bitmap holding depth (or range) information for each pixel.

disparity map a pixel bitmap holding binocular disparity information for each pixel. The information leads directly to a depth map.

field of view the angular extent of the observable world that is seen at any given moment (it is also called field of vision).

fixed pattern noise image sensor noise, constant offset for each pixel.

head-mounted display display device, worn on the head or as part of a helmet, that has a small display optic in front of one or each eye.

HSV color space HSL and HSV are two related representations of points in an RGB color space, which attempt to describe perceptual color relationships more accurately than RGB, while remaining computationally simple. HSL stands for hue, saturation, lightness, while HSV stands for hue, saturation, value.

image rectification the transformation process used to project multiple images onto a common image surface. It is used to correct a distorted image into a standard coordinate system.
motion blur  the apparent streaking of rapidly moving objects in a still image or a sequence of images such as a movie or animation
optical flow  the pattern of apparent motion of objects, surfaces, and edges in a visual scene caused by the relative motion between an observer (an eye or a camera) and the scene
pixel range, pixel depth  the range (or depth) of a pixel corresponds to its coordinate along the Z axis of the scene object it belongs to. The Z axis coincides with the optical axis of the camera, the origin is the camera.
segmentation map  a pixel bitmap holding segmentation information for each pixel, according to its membership to different classes. In this thesis, two classes are used, foreground and background.
vignetting  the reduction of an image’s brightness or saturation at the periphery compared to the image center
YUV 4:2:2  image encoding method where the two color components (U and V) are sampled at half the sample rate of the luminance (Y): the horizontal chroma resolution is halved. This reduces the bandwidth of a video signal by one-third with little to no visual difference.
A.3. Geometrical Model of the FingerMouse Stereo Camera Setup

A.3.1. Projective Stereo Geometry Model

In order to describe how (in geometrical terms) cameras acquire 2-dimensional images from the 3-dimensional real world, camera models are used. We will briefly describe such a model in stereo and derive some implications to our stereo systems.

The pinhole camera model describes the imaging as a projection of real world points onto the virtual image plane $P$, with the center of projection $C$. The projection of the real-world point $M$ to the image point $m$ can be seen as the intersection of the line through $M$ and $C$ and the image plane $P$.

![Figure A.1: The pinhole camera model](image)

In a real camera, the center of projection $C$ corresponds to the lens center, $f$ corresponds to the focal length. The virtual image plane corresponds to the surface of the image sensor, except that the real image plane is behind the center of projection, and is tilted upside down.

**Note.** The pinhole camera model is only an approximation of the real projection inside a given camera. The model is applicable when the focal distance is much smaller than the object’s distance to the camera. This is the case in our applications.
In order to write down the mathematical model of the perspective projection, which maps the real world coordinates \((X,Y,Z)\) to image coordinates \((u,v)\), we choose the following coordinate systems:

- the origin of the real world coordinates coincides with the center of projection \(C\).
- the origin of the image coordinates is the intersection of the image plane and the \(Z\) axis. (also called \textit{principal point})
- the \(Z\)-axis coincides with the optical axis (viewing direction) of the camera.
- the virtual image plane is parallel to the plane defined by the \(X\)-axis and \(Y\)-axis.
- the \(u\)-axis is parallel to the \(X\)-axis, the \(v\)-axis is parallel to the \(Y\)-axis.

In this case, the projection reads algebraically as follows:

\[
\begin{align*}
  u &= f \cdot \frac{X}{Z} \\
  v &= f \cdot \frac{Y}{Z}
\end{align*}
\]  

\textbf{Determining} \((X,Y,Z)\). In our application, \((u,v)\) is known from the measurement (image capture), and we want to determine \((X,Y,Z)\) (especially \(Z\), since it will help us to discriminate between foreground and background areas in the image).

It is clear that the knowledge of \(m(u,v)\) is not sufficient to determine the coordinates \((X,Y,Z)\) of \(M\), due to ambiguity. The possible positions of \(M\), when \(m\) is given, lie on the line defined by \(C\) and \(m\) (\(r\) in fig. A.2). This line is also called \textit{optical ray} of \(m\) (in analogy to all light falling in to the point \(m\)). Algebraically, the line is described by equation A.1, with \((u,v)\) set constant.
The ambiguity of $M$’s position can be resolved by having a second perspective projection with a different camera position, $m'$ being its projection to the second image plane (cf. fig. A.2).

An important property of $m'$ is that it may not appear anywhere in the second image plane, but only on a specific line: the projection of the optical ray of $m$ onto the second image plane. This projection is also a line, and is called the epipolar line ($e$ in fig. A.2). This property is important in case $m'$ has to be searched: it decreases the search space from the 2-dimensional image plane to the 1-dimensional epipolar line.

Once both projections $m$ and $m'$ of the point $M$ are known, the position of $M$ is determined: it is the intersection of the respective optical rays originating from $m$ and $m'$. The search for $m'$ corresponding to $m$ (and thus to $M$) is called the correspondence search. Methods to perform the search are presented in 3.2.

**Parallel cameras.** We now consider the following setup of two cameras (cf. A.3):

- 2 identical cameras
- image planes $P$ and $P'$ lie in the same plane
- the cameras translated on the X-axis by the distance $b$ (called camera baseline)

Using the pinhole camera models, we can write the coordinates of $m$ and $m'$ in each of the cameras’ coordinate systems, as seen in fig. A.3:
Figure A.3: Parallel camera setup

\[
\begin{align*}
&\begin{aligned}
&u = f \cdot \frac{X}{Z} \\
&v = f \cdot \frac{Y}{Z}
\end{aligned}
\quad \text{and} \quad \\
&\begin{aligned}
&u' = f \cdot \frac{X'}{Z'} \\
&v' = f \cdot \frac{Y'}{Z'}
\end{aligned}
\end{align*}
\]

(A.2)

The real-world coordinate system \((X',Y',Z')\) of the second camera model can be translated into the coordinate system of the first camera:

\[
\begin{align*}
&\begin{aligned}
&X' = X + b \\
&Y' = Y \\
&Z' = Z
\end{aligned}
\end{align*}
\]

(A.3)

This leads to an equation system, the allows to resolve \(M(X,Y,Z)\), for a given \(m(u,v)\) and \(m'(u',v')\):

\[
\begin{align*}
&\begin{aligned}
&u = f \cdot \frac{X}{Z} \\
&v = f \cdot \frac{Y}{Z}
\end{aligned}
\quad \text{and} \quad \\
&\begin{aligned}
&u' = f \cdot \frac{X+b}{Z} \\
&v' = f \cdot \frac{Y}{Z}
\end{aligned}
\end{align*}
\]

(A.4)

An important observation is that the vertical position \(v'\) of \(m'\) equals \(v\): the epipolar line in the case of the parallel cameras coincides
with the horizontal lines (parallel to the u-axis) in the image plane. The formulas reveal that the projection $m$ of $M$ onto the right camera is located further to the left, and vice versa (since $b > 0$).

The two expressions in A.4 can also be seen as the algebraic equations for the optical rays of $m$ and $m'$. The intersection is the point $M$, satisfying the 4 equations in (A.4), and its coordinates are:

$$\begin{cases} 
X = b \cdot \frac{u}{u'-u} \\
Y = b \cdot \frac{v}{u'-u} \\
Z = b \cdot f \cdot \frac{1}{u'-u}
\end{cases}$$

(A.5)

$(u' - u)$ expresses the horizontal offset between the projections of $M$ onto the two image planes. It is called disparity and we denote $d = u' - u$. The value $Z$ is referred to as the depth or range.

The above formulas use the same length units in all coordinate systems, e.g. meters. When analyzing digital images, pixels are normally used as units. With pixels sized $k_x \times k_y$ (width $\times$ height, [m]$\times$[m]), we introduce the x-y pixel coordinate system for the image, which translates as follows:

$$\begin{cases} 
x = u \cdot \frac{1}{k_x} \\
y = v \cdot \frac{1}{k_y}
\end{cases}$$

(A.6)

This leads us finally to the interrelation between the disparity $d$ [pixels] and the depth $Z$ [m] of a pixel:

$$Z = (b \cdot f \cdot \frac{X_i}{x_s}) \cdot \frac{1}{d}$$

(A.7)

$b =$ baseline; $f =$ focal length; $X_i =$ horizontal image resolution in pixels; $x_s =$ width of the image sensor; $(k_x = \frac{x_s}{X_i})$

Equation A.7 is used in the thesis, as equation 3.1 on page 23.
A.4. Additional Experimental Images and Results from Chapter 4

A.4.1. Sets 1 and 2

Table A.1 and table A.2 shows thee classification ratings for the sets 1 and 2.

The sets are shown in figures A.4 to A.23.

**Table A.1:** System output evaluated for the set 1

<table>
<thead>
<tr>
<th>Pair, $S_{filtered}$: $S_{color-combined}$:</th>
<th>$R_{fg}$ [%]</th>
<th>$R_{bg}$ [%]</th>
<th>$SNR_{fg-d}$</th>
<th>$\Delta_{cog}$ [pixels]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pair 1, $S_{filtered}$: $S_{color-combined}$:</td>
<td>69.69</td>
<td>99.48</td>
<td>133.70</td>
<td>2.40</td>
</tr>
<tr>
<td></td>
<td>96.57</td>
<td>99.86</td>
<td>714.03</td>
<td>0.70</td>
</tr>
<tr>
<td>Pair 2, $S_{filtered}$: $S_{color-combined}$:</td>
<td>73.08</td>
<td>99.50</td>
<td>146.86</td>
<td>5.54</td>
</tr>
<tr>
<td></td>
<td>97.41</td>
<td>99.55</td>
<td>215.94</td>
<td>2.12</td>
</tr>
<tr>
<td>Pair 3, $S_{filtered}$: $S_{color-combined}$:</td>
<td>84.79</td>
<td>99.50</td>
<td>168.87</td>
<td>4.71</td>
</tr>
<tr>
<td></td>
<td>99.06</td>
<td>99.83</td>
<td>588.90</td>
<td>2.69</td>
</tr>
<tr>
<td>Pair 4, $S_{filtered}$: $S_{color-combined}$:</td>
<td>76.13</td>
<td>99.61</td>
<td>193.14</td>
<td>9.61</td>
</tr>
<tr>
<td></td>
<td>97.66</td>
<td>99.83</td>
<td>569.04</td>
<td>3.23</td>
</tr>
<tr>
<td>Pair 5, $S_{filtered}$: $S_{color-combined}$:</td>
<td>76.07</td>
<td>99.24</td>
<td>100.15</td>
<td>1.87</td>
</tr>
<tr>
<td></td>
<td>98.17</td>
<td>99.79</td>
<td>462.03</td>
<td>2.69</td>
</tr>
<tr>
<td>Pair 6, $S_{filtered}$: $S_{color-combined}$:</td>
<td>77.72</td>
<td>99.45</td>
<td>141.73</td>
<td>0.26</td>
</tr>
<tr>
<td></td>
<td>98.56</td>
<td>99.54</td>
<td>213.85</td>
<td>3.17</td>
</tr>
<tr>
<td>Pair 7, $S_{filtered}$: $S_{color-combined}$:</td>
<td>80.15</td>
<td>99.75</td>
<td>318.43</td>
<td>6.14</td>
</tr>
<tr>
<td></td>
<td>97.14</td>
<td>99.85</td>
<td>658.73</td>
<td>2.71</td>
</tr>
<tr>
<td>Pair 8, $S_{filtered}$: $S_{color-combined}$:</td>
<td>68.87</td>
<td>99.53</td>
<td>147.92</td>
<td>3.20</td>
</tr>
<tr>
<td></td>
<td>97.12</td>
<td>99.91</td>
<td>1137.79</td>
<td>2.16</td>
</tr>
<tr>
<td>Pair 9, $S_{filtered}$: $S_{color-combined}$:</td>
<td>68.60</td>
<td>99.75</td>
<td>272.28</td>
<td>2.65</td>
</tr>
<tr>
<td></td>
<td>97.41</td>
<td>99.76</td>
<td>402.08</td>
<td>0.80</td>
</tr>
<tr>
<td>Pair 10, $S_{filtered}$: $S_{color-combined}$:</td>
<td>79.40</td>
<td>99.84</td>
<td>498.46</td>
<td>4.57</td>
</tr>
<tr>
<td></td>
<td>97.96</td>
<td>99.16</td>
<td>116.50</td>
<td>4.56</td>
</tr>
</tbody>
</table>
Table A.2: *System output evaluated for the set 2*

<table>
<thead>
<tr>
<th></th>
<th>$R_{fg}$ [%]</th>
<th>$R_{bg}$ [%]</th>
<th>$SNR_{fg-d}$</th>
<th>$\Delta_{cog}$ [pixels]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pair 1, $S_{filtered}$:</td>
<td>80.33</td>
<td>99.50</td>
<td>161.28</td>
<td>1.83</td>
</tr>
<tr>
<td>Pair 2, $S_{filtered}$:</td>
<td>81.88</td>
<td>99.13</td>
<td>94.64</td>
<td>0.87</td>
</tr>
<tr>
<td>Pair 3, $S_{filtered}$:</td>
<td>79.68</td>
<td>99.39</td>
<td>130.28</td>
<td>2.02</td>
</tr>
<tr>
<td>Pair 4, $S_{filtered}$:</td>
<td>80.26</td>
<td>99.36</td>
<td>125.48</td>
<td>5.59</td>
</tr>
<tr>
<td>Pair 5, $S_{filtered}$:</td>
<td>81.63</td>
<td>99.01</td>
<td>82.58</td>
<td>4.03</td>
</tr>
<tr>
<td>Pair 6, $S_{filtered}$:</td>
<td>80.33</td>
<td>99.70</td>
<td>266.80</td>
<td>3.01</td>
</tr>
<tr>
<td>Pair 7, $S_{filtered}$:</td>
<td>83.31</td>
<td>99.37</td>
<td>131.69</td>
<td>1.16</td>
</tr>
<tr>
<td>Pair 8, $S_{filtered}$:</td>
<td>82.24</td>
<td>98.93</td>
<td>77.04</td>
<td>1.74</td>
</tr>
<tr>
<td>Pair 9, $S_{filtered}$:</td>
<td>83.36</td>
<td>99.80</td>
<td>406.67</td>
<td>0.39</td>
</tr>
<tr>
<td>Pair 10, $S_{filtered}$:</td>
<td>76.57</td>
<td>99.54</td>
<td>165.33</td>
<td>7.55</td>
</tr>
</tbody>
</table>

Figure A.4: *Set 1, Pair 1*

Figure A.5: *Set 1, Pair 2*
Figure A.6: Set 1, Pair 3

Figure A.7: Set 1, Pair 4

Figure A.8: Set 1, Pair 5

Figure A.9: Set 1, Pair 6
A.4. Additional Experimental Images and Results from Chapter 4

Figure A.10: Set 1, Pair 7

Figure A.11: Set 1, Pair 8

Figure A.12: Set 1, Pair 9

Figure A.13: Set 1, Pair 10
Figure A.14: Set 2, Pair 1

Figure A.15: Set 2, Pair 2

Figure A.16: Set 2, Pair 3

Figure A.17: Set 2, Pair 4
A.4. Additional Experimental Images and Results from Chapter 4

Figure A.18: Set 2, Pair 5

Figure A.19: Set 2, Pair 6

Figure A.20: Set 2, Pair 7

Figure A.21: Set 2, Pair 8
Figure A.22: Set 2, Pair 9

Figure A.23: Set 2, Pair 10
### A.4.2. Numerical Results of Multi-Baseline-Focal Length Measurements

Tables A.4 and A.5 show the foreground classification results for $S_{filtered}$ and $S_{color-combined}$ in the multi-baseline-focal length experiment from section 4.3. The cells contain the values $R_{fg}$, $R_{bg}$, $SNR_{fg-d}$ and $\Delta_{cog}$ as indicated in table A.3. Different columns express different focal lengths $f$, different rows contain the results for the different baselines $b$. All $b$ and $f$ values are expressed in mm. The $f-b$ combinations that were not measured due to a too large $K$ are indicated as n.m.

**Table A.3: Ordering of values in tables A.4 and A.5**

<table>
<thead>
<tr>
<th>$R_{fg}$ [%]</th>
<th>$R_{bg}$ [%]</th>
<th>$SNR_{fg-d}$</th>
<th>$\Delta_{cog}$ [pixels]</th>
</tr>
</thead>
</table>


### Table A.4: Multi-Baseline-Focal Length Experiment, $S_{filtered}$

<table>
<thead>
<tr>
<th>$b$</th>
<th>18.6</th>
<th>23.6</th>
<th>28.6</th>
<th>33.6</th>
<th>38.6</th>
<th>43.6</th>
<th>48.6</th>
<th>53.6</th>
<th>58.6</th>
<th>63.6</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.8</td>
<td>76.4</td>
<td>96.6</td>
<td>5.5</td>
<td>55.0</td>
<td>91.9</td>
<td>4.8</td>
<td>34.0</td>
<td>80.1</td>
<td>2.5</td>
</tr>
<tr>
<td></td>
<td>99.9</td>
<td>99.9</td>
<td>99.6</td>
<td>100.0</td>
<td>99.9</td>
<td>99.6</td>
<td>100.0</td>
<td>99.9</td>
<td>99.8</td>
<td>100.0</td>
</tr>
<tr>
<td>12</td>
<td>789</td>
<td>257</td>
<td>138</td>
<td>479</td>
<td>252</td>
<td>175</td>
<td>393</td>
<td>407</td>
<td>109</td>
<td></td>
</tr>
<tr>
<td>42.3</td>
<td>8.4</td>
<td>0.3</td>
<td>37.7</td>
<td>11.9</td>
<td>2.7</td>
<td>32.4</td>
<td>7.3</td>
<td>2.5</td>
<td>10.9</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>79.3</td>
<td>23.4</td>
<td>90.5</td>
<td>61.4</td>
<td>25.0</td>
<td>82.1</td>
<td>40.7</td>
<td>14.2</td>
<td>72.0</td>
<td>27.3</td>
</tr>
<tr>
<td></td>
<td>100.0</td>
<td>100.0</td>
<td>99.8</td>
<td>99.9</td>
<td>99.9</td>
<td>99.8</td>
<td>99.9</td>
<td>99.8</td>
<td>99.8</td>
<td>100.0</td>
</tr>
<tr>
<td>2331</td>
<td>0</td>
<td>512</td>
<td>587</td>
<td>386</td>
<td>439</td>
<td>483</td>
<td>229</td>
<td>407</td>
<td>568</td>
<td></td>
</tr>
<tr>
<td>8.0</td>
<td>0</td>
<td>1.1</td>
<td>13.5</td>
<td>44.9</td>
<td>3.6</td>
<td>10.2</td>
<td>42.2</td>
<td>4.1</td>
<td>25.2</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>91.4</td>
<td>91.5</td>
<td>89.7</td>
<td>84.5</td>
<td>84.5</td>
<td>86.5</td>
<td>89.6</td>
<td>88.7</td>
<td>87.4</td>
<td>85.5</td>
</tr>
<tr>
<td></td>
<td>99.8</td>
<td>99.8</td>
<td>99.7</td>
<td>99.7</td>
<td>99.8</td>
<td>99.7</td>
<td>99.9</td>
<td>99.8</td>
<td>99.8</td>
<td>99.8</td>
</tr>
<tr>
<td>588</td>
<td>384</td>
<td>288</td>
<td>306</td>
<td>506</td>
<td>340</td>
<td>657</td>
<td>459</td>
<td>398</td>
<td>554</td>
<td></td>
</tr>
<tr>
<td>0.7</td>
<td>1.3</td>
<td>2.5</td>
<td>1.8</td>
<td>2.7</td>
<td>2.9</td>
<td>1.3</td>
<td>1.4</td>
<td>1.4</td>
<td>2.5</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>84.8</td>
<td>87.0</td>
<td>91.3</td>
<td>80.5</td>
<td>82.5</td>
<td>77.2</td>
<td>78.1</td>
<td>69.6</td>
<td>70.3</td>
<td>71.6</td>
</tr>
<tr>
<td></td>
<td>99.9</td>
<td>99.8</td>
<td>99.8</td>
<td>99.8</td>
<td>99.8</td>
<td>99.8</td>
<td>99.8</td>
<td>99.8</td>
<td>99.8</td>
<td>99.8</td>
</tr>
<tr>
<td>997</td>
<td>450</td>
<td>509</td>
<td>326</td>
<td>345</td>
<td>326</td>
<td>444</td>
<td>312</td>
<td>366</td>
<td>497</td>
<td></td>
</tr>
<tr>
<td>0.2</td>
<td>2.0</td>
<td>1.0</td>
<td>4.2</td>
<td>1.5</td>
<td>5.3</td>
<td>4.2</td>
<td>1.4</td>
<td>5.8</td>
<td>2.7</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>86.0</td>
<td>87.5</td>
<td>84.5</td>
<td>81.5</td>
<td>78.1</td>
<td>76.7</td>
<td>75.0</td>
<td>71.7</td>
<td>70.3</td>
<td>66.7</td>
</tr>
<tr>
<td></td>
<td>99.9</td>
<td>99.9</td>
<td>99.8</td>
<td>99.8</td>
<td>99.9</td>
<td>99.8</td>
<td>99.9</td>
<td>99.9</td>
<td>99.9</td>
<td>99.8</td>
</tr>
<tr>
<td>770</td>
<td>784</td>
<td>492</td>
<td>394</td>
<td>580</td>
<td>410</td>
<td>550</td>
<td>491</td>
<td>626</td>
<td>400</td>
<td></td>
</tr>
<tr>
<td>1.4</td>
<td>2.3</td>
<td>3.6</td>
<td>3.3</td>
<td>4.5</td>
<td>4.4</td>
<td>3.8</td>
<td>1.6</td>
<td>2.8</td>
<td>3.1</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>87.5</td>
<td>82.5</td>
<td>80.4</td>
<td>79.1</td>
<td>77.5</td>
<td>76.3</td>
<td>74.0</td>
<td>67.7</td>
<td>61.8</td>
<td>57.1</td>
</tr>
<tr>
<td></td>
<td>99.9</td>
<td>99.8</td>
<td>99.8</td>
<td>99.8</td>
<td>99.8</td>
<td>99.8</td>
<td>99.8</td>
<td>99.9</td>
<td>99.9</td>
<td>100.0</td>
</tr>
<tr>
<td>679</td>
<td>528</td>
<td>461</td>
<td>382</td>
<td>457</td>
<td>340</td>
<td>383</td>
<td>463</td>
<td>918</td>
<td>1254</td>
<td></td>
</tr>
<tr>
<td>2.4</td>
<td>2.7</td>
<td>3.1</td>
<td>4.0</td>
<td>4.7</td>
<td>4.6</td>
<td>3.4</td>
<td>2.5</td>
<td>3.7</td>
<td>4.0</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>87.4</td>
<td>84.6</td>
<td>81.8</td>
<td>80.1</td>
<td>75.7</td>
<td>74.1</td>
<td>72.9</td>
<td>69.2</td>
<td>54.7</td>
<td>n.m</td>
</tr>
<tr>
<td></td>
<td>99.9</td>
<td>99.8</td>
<td>99.8</td>
<td>99.7</td>
<td>99.9</td>
<td>99.8</td>
<td>99.8</td>
<td>99.9</td>
<td>99.9</td>
<td>100.0</td>
</tr>
<tr>
<td>706</td>
<td>525</td>
<td>357</td>
<td>284</td>
<td>515</td>
<td>425</td>
<td>426</td>
<td>1075</td>
<td>3135</td>
<td>n.m</td>
<td></td>
</tr>
<tr>
<td>1.6</td>
<td>1.5</td>
<td>2.2</td>
<td>2.7</td>
<td>1.5</td>
<td>1.9</td>
<td>3.4</td>
<td>3.0</td>
<td>6.2</td>
<td>n.m</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>84.6</td>
<td>80.5</td>
<td>79.1</td>
<td>75.3</td>
<td>73.4</td>
<td>61.2</td>
<td>13.6</td>
<td>n.m</td>
<td>n.m</td>
<td>n.m</td>
</tr>
<tr>
<td></td>
<td>99.9</td>
<td>99.8</td>
<td>99.8</td>
<td>99.7</td>
<td>99.9</td>
<td>99.9</td>
<td>99.9</td>
<td>99.9</td>
<td>152</td>
<td></td>
</tr>
<tr>
<td>590</td>
<td>492</td>
<td>331</td>
<td>283</td>
<td>535</td>
<td>463</td>
<td>152</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.9</td>
<td>2.0</td>
<td>2.6</td>
<td>2.4</td>
<td>2.5</td>
<td>5.9</td>
<td>15.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>45</td>
<td>75.3</td>
<td>54.9</td>
<td>n.m</td>
<td>n.m</td>
<td>n.m</td>
<td>n.m</td>
<td>n.m</td>
<td>n.m</td>
<td>n.m</td>
<td></td>
</tr>
<tr>
<td>99.9</td>
<td>99.9</td>
<td>1476</td>
<td>963</td>
<td>2.0</td>
<td>5.1</td>
<td>n.m</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>64</td>
<td>5.5</td>
<td>100.0</td>
<td>2260</td>
<td>3.76</td>
<td>n.m</td>
<td>n.m</td>
<td>n.m</td>
<td>n.m</td>
<td>n.m</td>
</tr>
</tbody>
</table>

*Table A.4: Multi-Baseline-Focal Length Experiment, $S_{filtered}$*
Table A.5: Multi-Baseline-Focal Length Experiment, $S_{\text{color-combined}}$

<table>
<thead>
<tr>
<th>$b \backslash f$</th>
<th>18.6</th>
<th>23.6</th>
<th>28.6</th>
<th>33.6</th>
<th>38.6</th>
<th>43.6</th>
<th>48.6</th>
<th>53.6</th>
<th>58.6</th>
<th>63.6</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>4.7</td>
<td>93.6</td>
<td>100.0</td>
<td>17.8</td>
<td>83.1</td>
<td>99.5</td>
<td>16.2</td>
<td>68.5</td>
<td>98.1</td>
<td>9.7</td>
</tr>
<tr>
<td></td>
<td>99.7</td>
<td>98.8</td>
<td>97.5</td>
<td>99.8</td>
<td>99.1</td>
<td>97.8</td>
<td>99.9</td>
<td>99.3</td>
<td>98.0</td>
<td>99.9</td>
</tr>
<tr>
<td>19</td>
<td>76</td>
<td>40</td>
<td>92</td>
<td>95</td>
<td>46</td>
<td>198</td>
<td>102</td>
<td>50</td>
<td>142</td>
<td></td>
</tr>
<tr>
<td>32.5</td>
<td>0.6</td>
<td>3.5</td>
<td>33.9</td>
<td>1.4</td>
<td>2.6</td>
<td>25.0</td>
<td>6.1</td>
<td>2.8</td>
<td>23.7</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>90.8</td>
<td>40.8</td>
<td>99.7</td>
<td>83.0</td>
<td>37.0</td>
<td>98.6</td>
<td>69.6</td>
<td>22.6</td>
<td>95.7</td>
<td>59.6</td>
</tr>
<tr>
<td></td>
<td>99.2</td>
<td>100.0</td>
<td>98.0</td>
<td>99.3</td>
<td>99.9</td>
<td>98.3</td>
<td>99.4</td>
<td>99.8</td>
<td>98.6</td>
<td>99.4</td>
</tr>
<tr>
<td>112</td>
<td>16461</td>
<td>51</td>
<td>119</td>
<td>270</td>
<td>58</td>
<td>112</td>
<td>136</td>
<td>67</td>
<td>108</td>
<td></td>
</tr>
<tr>
<td>3.6</td>
<td>24.3</td>
<td>2.8</td>
<td>3.6</td>
<td>34.1</td>
<td>1.7</td>
<td>2.2</td>
<td>20.9</td>
<td>3.6</td>
<td>17.0</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>99.8</td>
<td>99.9</td>
<td>100.0</td>
<td>99.9</td>
<td>99.6</td>
<td>99.9</td>
<td>99.8</td>
<td>99.8</td>
<td>99.8</td>
<td>99.7</td>
</tr>
<tr>
<td></td>
<td>98.3</td>
<td>98.1</td>
<td>97.9</td>
<td>98.0</td>
<td>98.5</td>
<td>98.4</td>
<td>98.5</td>
<td>98.5</td>
<td>98.3</td>
<td></td>
</tr>
<tr>
<td>58</td>
<td>53</td>
<td>47</td>
<td>51</td>
<td>68</td>
<td>61</td>
<td>67</td>
<td>62</td>
<td>65</td>
<td>57</td>
<td></td>
</tr>
<tr>
<td>3.8</td>
<td>4.0</td>
<td>3.5</td>
<td>2.5</td>
<td>2.3</td>
<td>1.6</td>
<td>1.7</td>
<td>1.7</td>
<td>2.2</td>
<td>1.5</td>
<td>1.5</td>
</tr>
<tr>
<td>8</td>
<td>99.7</td>
<td>99.3</td>
<td>99.7</td>
<td>99.0</td>
<td>99.2</td>
<td>98.0</td>
<td>98.6</td>
<td>97.4</td>
<td>96.8</td>
<td>96.8</td>
</tr>
<tr>
<td></td>
<td>98.6</td>
<td>98.4</td>
<td>98.0</td>
<td>98.4</td>
<td>98.5</td>
<td>98.5</td>
<td>98.5</td>
<td>98.6</td>
<td>98.7</td>
<td>98.4</td>
</tr>
<tr>
<td>69</td>
<td>63</td>
<td>49</td>
<td>62</td>
<td>66</td>
<td>64</td>
<td>64</td>
<td>68</td>
<td>73</td>
<td>60</td>
<td></td>
</tr>
<tr>
<td>2.4</td>
<td>2.3</td>
<td>3.0</td>
<td>1.6</td>
<td>1.7</td>
<td>0.8</td>
<td>0.7</td>
<td>0.7</td>
<td>0.6</td>
<td>1.4</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>99.2</td>
<td>99.3</td>
<td>99.4</td>
<td>99.1</td>
<td>98.6</td>
<td>97.9</td>
<td>98.2</td>
<td>97.4</td>
<td>97.6</td>
<td>95.7</td>
</tr>
<tr>
<td></td>
<td>98.4</td>
<td>98.5</td>
<td>98.0</td>
<td>98.1</td>
<td>98.5</td>
<td>98.6</td>
<td>98.5</td>
<td>98.5</td>
<td>98.5</td>
<td></td>
</tr>
<tr>
<td>63</td>
<td>67</td>
<td>49</td>
<td>52</td>
<td>68</td>
<td>68</td>
<td>67</td>
<td>70</td>
<td>65</td>
<td>62</td>
<td></td>
</tr>
<tr>
<td>3.2</td>
<td>2.4</td>
<td>3.7</td>
<td>3.5</td>
<td>2.1</td>
<td>0.7</td>
<td>1.2</td>
<td>0.7</td>
<td>1.9</td>
<td>2.1</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>99.2</td>
<td>99.0</td>
<td>99.0</td>
<td>99.0</td>
<td>98.8</td>
<td>98.4</td>
<td>97.5</td>
<td>96.4</td>
<td>94.4</td>
<td>91.0</td>
</tr>
<tr>
<td></td>
<td>98.5</td>
<td>98.4</td>
<td>98.1</td>
<td>98.2</td>
<td>98.7</td>
<td>98.6</td>
<td>98.6</td>
<td>98.7</td>
<td>98.7</td>
<td>98.8</td>
</tr>
<tr>
<td>66</td>
<td>60</td>
<td>53</td>
<td>56</td>
<td>77</td>
<td>69</td>
<td>68</td>
<td>72</td>
<td>73</td>
<td>76</td>
<td></td>
</tr>
<tr>
<td>2.5</td>
<td>2.1</td>
<td>2.5</td>
<td>2.0</td>
<td>1.3</td>
<td>0.6</td>
<td>0.5</td>
<td>1.1</td>
<td>1.7</td>
<td>2.3</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>99.7</td>
<td>99.8</td>
<td>99.5</td>
<td>99.7</td>
<td>98.9</td>
<td>98.7</td>
<td>98.3</td>
<td>96.8</td>
<td>86.0</td>
<td>n.m</td>
</tr>
<tr>
<td></td>
<td>98.6</td>
<td>98.3</td>
<td>98.0</td>
<td>97.9</td>
<td>98.5</td>
<td>98.8</td>
<td>98.9</td>
<td>99.0</td>
<td>99.1</td>
<td></td>
</tr>
<tr>
<td>69</td>
<td>59</td>
<td>49</td>
<td>48</td>
<td>67</td>
<td>80</td>
<td>93</td>
<td>101</td>
<td>98</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.7</td>
<td>2.7</td>
<td>2.8</td>
<td>2.8</td>
<td>2.3</td>
<td>0.8</td>
<td>0.3</td>
<td>0.8</td>
<td>5.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>99.6</td>
<td>99.3</td>
<td>99.4</td>
<td>99.5</td>
<td>98.7</td>
<td>92.1</td>
<td>33.4</td>
<td>n.m</td>
<td>n.m</td>
<td>n.m</td>
</tr>
<tr>
<td></td>
<td>98.5</td>
<td>98.3</td>
<td>98.1</td>
<td>98.2</td>
<td>98.8</td>
<td>98.9</td>
<td>99.5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>67</td>
<td>58</td>
<td>51</td>
<td>55</td>
<td>83</td>
<td>83</td>
<td>64</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.0</td>
<td>2.9</td>
<td>2.8</td>
<td>2.4</td>
<td>1.6</td>
<td>4.5</td>
<td>12.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>45</td>
<td>99.0</td>
<td>98.7</td>
<td>98.7</td>
<td>69</td>
<td>5.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>98.6</td>
<td>98.7</td>
<td>69</td>
<td>5.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>13.3</td>
<td>n.m</td>
<td>n.m</td>
<td>n.m</td>
<td>n.m</td>
<td>n.m</td>
<td>n.m</td>
<td>n.m</td>
<td>n.m</td>
<td>n.m</td>
</tr>
<tr>
<td>55</td>
<td>99.8</td>
<td>n.m</td>
<td>n.m</td>
<td>n.m</td>
<td>n.m</td>
<td>n.m</td>
<td>n.m</td>
<td>n.m</td>
<td>n.m</td>
<td>n.m</td>
</tr>
<tr>
<td>26.6</td>
<td>99.8</td>
<td>n.m</td>
<td>n.m</td>
<td>n.m</td>
<td>n.m</td>
<td>n.m</td>
<td>n.m</td>
<td>n.m</td>
<td>n.m</td>
<td>n.m</td>
</tr>
</tbody>
</table>
A.5. Core Voltage for Different IC Clock Frequencies $f_{IC}$

In order to determine the core voltage range for a given IC clock frequency $f_{IC}$, IC functionality tests were conducted at different core voltages and operating frequencies. The results are shown in figure A.24.

![Core Voltage for Different IC Clock Frequencies](image)

**Figure A.24:** Functionality (fail/pass) of the FingerMouse IC at different core voltages (y-axis) and operating frequencies (clock period on x-axis)

Higher operating frequencies require a higher core voltage for the IC to function. The tests show IC functionality up to a $f_{IC} = 166$ MHz ($\tau_{IC} = 6$ ns, $V_{DD} = 4.4$ V), or core voltages as low as 1.4 V ($\tau_{IC} = 19$ ns, $f_{IC} = 5.25$ MHz).

A.6. Scalability Considerations of the FingerMouse IC Architecture

**Frequency $f_{IC}$ scaling**

As a completely synchronous circuit, an increase of the operating frequency by a factor $k$ leads to all operations being carried out faster by a factor $k$. The new pixel rate $f_{pixel}$ of the system will be $k \times 5$ Mpixel/s.
The pixel rate is the product of spatial and temporal resolution:

\[ f_{\text{pixel}} = X_i \cdot Y_i \cdot f_{\text{frame}} \quad (A.8) \]

One or more of the three resolutions can be increased following equation A.8. The disparity range cannot be increased by \( f_{\text{IC}} \) scaling, as it is hard wired in the circuit. For the same reason, the maximum horizontal resolution \( X_{i,\text{max}} = 340 \) pixels remains.

**Chip Size (Area) \( A_{\text{IC}} \) scaling**

The pixel rate of the system can also be increased by enlarging the circuit, adding more logic units and buffers. More parallel operating units will lead to the higher processing speed.

In order to scale the pixel rate \( f_{\text{pixel}} \) by a factor \( k \), the number of block comparisons per second has to increase by \( k \), requiring the following:

- the bandwidth of the interface between intensity row buffer SRAM and search block ring buffers (currently 192 bits per \( \tau_{\text{IC}} \)) needs to be scaled by \( k \)
- the size of the search block ring buffers has to be increased to buffer \( k \times 4 \) candidate blocks
- the number of parallel \( SAD \) and \( census \) block comparison units (currently 16) has to be increased by the factor \( k \)
- the segmentation processing layer needs to process pixels faster by a factor \( k \). Currently, it processes pixels at \( f_{\text{pixel}} = \frac{1}{16} f_{\text{IC}} \). Without architectural changes, it can process at a higher pixel rate, up to \( f_{\text{pixel}} = f_{\text{IC}} \). A further speed increase will require several parallel segmentation processing units.

The increased pixel rate allows either a higher horizontal resolution \( X_i \), a higher vertical resolution \( Y_i \), a higher temporal resolution \( f_{\text{frame}} \) or a combination of the three, according to equation A.8.

The FingerMouse circuit can also be altered to achieve a higher disparity range. In order to scale \( d_{\text{limit}} \) by a factor \( k \), the following is required:

- the bandwidth of the interface between intensity row buffer SRAM and search block ring buffers (currently 192 bits per \( \tau_{\text{IC}} \)) needs to be scaled by \( k \)
• the size of the search block ring buffers has to be increased to buffer $k \times 4$ candidate blocks

• the number of parallel SAD and census block comparison units (currently 16) has to be increased by the factor $k$

• the size of the left-right fusion buffer (currently 48 bits) has to be increased by the factor $k$

In order to scale the maximum horizontal resolution $X_{i,\text{max}}$ by a factor $k$, without changing the pixel rate $f_{\text{pixel}}$, it is necessary to:

• the size of the intensity row buffer and the color row buffer has to be increased by the factor $k$

The presented area scaling methods all require a part of the circuit to be multiplied by the factor $k$, while another part of the circuit is invariant. This corresponds to a circuit increase linear with $k$, leading to the IC area $A_{\text{IC}}$ and power consumption $P_{\text{IC}}$ to increase linear with $k$.

A.7. Description of the Earlier FingerMouse Prototypes

A.7.1. The DSP-FingerMouse

The DSP-based prototype was the first FingerMouse system implemented in the project, and was developed in 2001/2002 (cf. figure A.25).

It uses a DSP (TI TMS320VC33, @75 MHz) as the processing unit, with 64 KBytes of integrated memory. The system uses two black & white CMOS image sensors of $256 \times 256$ resolution (APS256D by CSEM, a low power sensor prototype [121]) and a distance sensor (Sharp GP2D02).

The cameras are triggered synchronously, but the output timing is not deterministic: the two camera signals showed to have a phase of up to 5 ms (5000 pixels). A buffer is required to equalize this phase. On the other hand, the image transmission from the cameras to the DSP is implemented as a software interface, polling incoming pixels. This interface, optimized in assembler code, is just fast enough to transfer the image in real-time. Therefore, a stereo frame buffer is realized, and the image processing algorithm is carried out sequentially. Since the
A.7. Description of the Earlier FingerMouse Prototypes

Figure A.25: The DSP-FingerMouse prototype
(a) front side, next to credit card, the distance sensor is seen in the middle
(b) backside

internal RAM size is limited to 64 Kbytes, the images are downsampled on-the-fly to 128 × 128 resolution.

The transfer time of the images is 65ms (256 × 256 transmitted at 1 Mpixel/s). Subsequently, the difference image is processed and the global threshold computed concurrently (∼245,000 instruction cycles, 3.5ms), then the image is thresholded (∼135,000 cycles, 1.8ms) and finally the erosion is performed with concurrent center-of-gravity computation (∼610,000 cycles, 8.2ms) and transferred (also sequentially, since implemented in software without UART) over RS232 (2ms @9600 bit/s). The total algorithmic processing time (13.5ms) is relatively small compared to the image transmission time (65ms), so that the sequential processing is not much slower than concurrent image transmission and processing schemes.

The achieved frame rate is 12.5 frames/s (0.25 M stereo pixel/s).

The work was presented at the 2002 UbiComp conference, cf. [60].

A.7.2. The FPGA-FingerMouse

The FPGA-FingerMouse system was developed in 2003/2004. It was designed to run the stereo-substraction algorithms and serve as a test-
The system uses an FPGA (Xilinx Spartan II) as a processing unit. A 16 Mbit SRAM chip serves as a buffer for several frames in full resolution. The employed image sensors, National LM9628, deliver full color images in raw bayer pattern, and can operate in a non-linear sensitivity mode, achieving a dynamic range of 110dB. The system further features a flash RAM for autonomous start up of the FPGA configuration, an interface to the Sharp GP2D02 distance sensor. A dynamic clock generation circuit allows the system to set its own operation clock rate for performance/consumption throttleing, between 3.7 MHz and 47 MHz (using an *ICS IC525-01* programmable clock generator). The embedded system is divided in a main module, holding the FPGA and the other components and a separate camera module. The system is seen in figure A.26.

The FPGA implementation allowed to parallelize operations, such as image transfer. The transfer between the image sensors and the FPGA is synchronized and the pixel streams have no phase, the FPGA providing the master clock to the cameras. This concept is later used in the ASIC-FingerMouse. While the software image interface in the DSP-FingerMouse posed a bottleneck to the image transmission speed, the new interface allows a throughput of 30 fps at 640×480 resolution.

The implementation also features a bayer pattern conversion. It was realized on-the-fly, without buffering the images into the SRAM. Instead a 3-image row buffer in FIFO principle was achieved (using FPGA internal RAM buffers), another concept later used in the ASIC-FingerMouse (not for color conversion, but as the pixel buffer to the block matching algorithm).

**A.8. Swissranger SR-3000 sample images**

Figure A.27 shows a hand captured by the *Swissranger SR-3000* time-of-flight camera.
Figure A.26: The FPGA-FingerMouse prototype
(a) prototype boards (next to a credit card)
(b) assembled prototype (two PCBs stacked), with optics and plastic case

Figure A.27: A depth map measured by the Swissranger SR-3000: a hand in the foreground is differentiated clearly by its range. The SR-3000 system’s output can be easily transformed to a foreground/background segmentation. (image courtesy: Mesa Imaging AG, Zürich, www.mesa-imaging.ch)
A.9. ITRS numbers

Table A.6 shows the trend numbers from the ITRS 2007 report ([87]) that were used to perform the FingerMouse $P_{IC}$ and $A_{IC}$ projections in figure 6.1 (p. 117). The trend numbers prior to the year 2007 were extracted from the ITRS 1999 report ([122]). The following equations were used to calculate $P_{IC}$ and $A_{IC}$:

\begin{align*}
P_{IC} & = 78 \cdot \frac{V_{DD}^2}{2.5^2} \ [mW] \quad \text{(A.9)} \\
A_{IC} & = \frac{0.38}{\text{density}_{\text{logic}}} + \frac{0.032768 \cdot 6}{\text{density}_{\text{SRAM}}} \ [mm^2] \quad \text{(A.10)}
\end{align*}

(The FingerMouse IC contains 0.38 million transistors plus 32768 SRAM cells, each with 6 transistors.)
### Table A.6: *ITRS 2007 trends*

<table>
<thead>
<tr>
<th>year</th>
<th>man. process [nm]</th>
<th>$V_{DD}$</th>
<th>density$_{logic}$</th>
<th>density$_{SRAM}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2007</td>
<td>65</td>
<td>0.8</td>
<td>1.54</td>
<td>8.27</td>
</tr>
<tr>
<td>2008</td>
<td>65</td>
<td>0.8</td>
<td>1.94</td>
<td>10.57</td>
</tr>
<tr>
<td>2009</td>
<td>50</td>
<td>0.8</td>
<td>2.45</td>
<td>13.48</td>
</tr>
<tr>
<td>2010</td>
<td>45</td>
<td>0.7</td>
<td>3.09</td>
<td>17.18</td>
</tr>
<tr>
<td>2011</td>
<td>45</td>
<td>0.7</td>
<td>3.89</td>
<td>21.87</td>
</tr>
<tr>
<td>2012</td>
<td>36</td>
<td>0.7</td>
<td>4.9</td>
<td>27.81</td>
</tr>
<tr>
<td>2013</td>
<td>32</td>
<td>0.6</td>
<td>6.17</td>
<td>35.32</td>
</tr>
<tr>
<td>2014</td>
<td>32</td>
<td>0.6</td>
<td>7.78</td>
<td>44.84</td>
</tr>
<tr>
<td>2015</td>
<td>25</td>
<td>0.6</td>
<td>9.8</td>
<td>56.87</td>
</tr>
<tr>
<td>2016</td>
<td>22</td>
<td>0.5</td>
<td>12.35</td>
<td>72.08</td>
</tr>
<tr>
<td>2017</td>
<td>22</td>
<td>0.5</td>
<td>15.55</td>
<td>91.30</td>
</tr>
<tr>
<td>2018</td>
<td>18</td>
<td>0.5</td>
<td>19.6</td>
<td>115.58</td>
</tr>
<tr>
<td>2019</td>
<td>16</td>
<td>0.5</td>
<td>24.69</td>
<td>146.25</td>
</tr>
<tr>
<td>2020</td>
<td>16</td>
<td>0.5</td>
<td>31.11</td>
<td>184.97</td>
</tr>
<tr>
<td>2021</td>
<td>16</td>
<td>0.45</td>
<td>39.2</td>
<td>233.94</td>
</tr>
<tr>
<td>2022</td>
<td>11</td>
<td>0.45</td>
<td>49.38</td>
<td>295.88</td>
</tr>
</tbody>
</table>

Table A.7: 1: manufacturing process trend number, DRAM product half pitch, rounded numbers (ITRS2007, table C, p. 61)  
2: supply voltage $V_{DD}$ (low operating power IC, high $V_{DD}$ transistors, ITRS2007, table 6a and 6b, p. 82)  
3: Transistor density, logic ($M$ transistors / mm$^2$), MPU cost-performance product generations (ITRS2007, table 1g and 1h, p. 70)  
4: Transistor density, SRAM ($M$ transistors / mm$^2$), MPU cost-performance product generations (ITRS2007, table 1g and 1h, p. 70)
Bibliography


[27] Griesser, A., Roeck, S.D., Neubeck, A., Gool, L.V.: Gpu-based foreground-background segmentation using an extended colinear-


the 10th IEEE International Symposium on Wearable Computers, ISWC’06 (2006)


[108] Chen, C., Tsai, H., Huang, K., Liu, H.: Study for cross contamination between CMOS image sensor and IC product. Advanced


