THESIS ON INFORMATICS AND SYSTEM ENGINEERING C55

The Tactile Feedback Device for Multi-Touch User Interfaces

ERKKI JOASOON

TUT
PRESS

TALLINN UNIVERSITY OF TECHNOLOGY
Faculty of Information Technology
Department of Informatics

Dissertation was accepted for the defence of the degree of Doctor of Philosophy in Informatics on May 12, 2010

Supervisor: Associate Professor Jaak Henno, PhD
Department of Informatics
Faculty of Information Technology
Tallinn University of Technology

Opponents: Professor Rein Laaneots, PhD
Department of Mechatronics
Faculty of Mechanical Engineering
Tallinn University of Technology

Professor Hannu Jaakkola, PhD
Tampere University of Technology, Pori

Defence of the thesis: June 21, 2010

Declaration:
Hereby I declare that this doctoral thesis, my original investigation and achievement, submitted for the doctoral degree at Tallinn University of Technology has not been submitted for any academic degree.

/Erkki Joasoon/

ISSN 1406-4731
ISBN 978-9949-23-009-9

INFORMAATIKA JA SÜSTEEMITEHNIKA C55

Kompeaistingul põhineva tagasisidega seade multipuutelise kasutajaliidese jaoks

ERKKI JOASOON

TTÜ
KIRJASTUS

Dedication

This thesis is dedicated to Harald Joasoon, my grandfather, who planted in me the seed of endless knowledge seeking...

The tactile feedback device for multi-touch user interfaces

Abstract

This research contributes to one of the most important fields in technology - the communication between man and computer. When the technology evolves and the computer systems become faster and there is more information available to access, we also need more efficient tools to manage it – better User Interfaces (UI). As a solution to this problem we propose implementation of additional channels of UI: multi-touch input with tactile feedback.

In this research we present a tactile feedback device – a multi finger input system that is enhanced with individual haptic feedback to each finger, and consider some issues that arise when this device is implemented. The aim of the research was to introduce simple, robust and low-cost UI device and technology that can quite easily be implemented on top of current, existing hardware.

Design of the introduced here tactile UI device was guided by the communication localization principle: communication, information exchange should include only information that human senses at are able to receive.

The main subject of this thesis, the tactile feedback device (TFD) for multi-touch user interfaces (MTUF) is a new technology with unique qualities. Uniqueness of this solution is verified and confirmed by the patent [Joasoon 2008-3] issued by Estonian Patent Office (EPA) on 15.12.2008 and it covers not only hardware but also the framework of how the hardware operates and works together with other systems.

One of problems connected with multi finger input system is positioning of fingers - a problem which also arises in other types of multi-touch systems e.g. which use Frustrated Total Internal Reflection (FTIR) or Rear Diffused Illumination (Rear DI) technology. This problem becomes very serious when the size of a multi-touch screen becomes bigger. The most common solution to finger positioning is use of video cameras. For precise positioning, the cameras should have good resolution and their video feed has to be processed in real time. However, when the camera resolution increases, the video feed bit rate grows quickly thus entailing the need for a lot more CPU power to process it. Here is developed the Limited Trackframe method which provides a scalable solution to that problem and is presented in the paper [Joasoon 2008-2].

This subtheme is linked to the main subject in several ways:

- The multi-touch technology (MTT) is one of the main enabling technologies for the TDF, therefore solving MTT problems will definitely make the road for all MTUF technologies easier.
- The basic approach to solve the MTT problem - localization of information exchange - is the same that was used to develop TDF technology.

This research opens also several directions for the future researches in the human computer interface, like combining dual-level positioning with blob tracking optimisation and using detailed tactile feedback also in other systems, besides computer interfaces (more detail description is in Chapter 6.2).

Acknowledgements

I would like to thank my supervisor Associate Professor Jaak Henno for those five years of support and continuous encouragement that made this thesis possible.

I want to thank Professor Rein Kuusik for believing in me and for the kind words and support on my difficult moments.

I thank my good friends Jüri Põldre, Ago Kuusik and Madis Listak for giving me the initial push and inspiration to stand up and start this process of PhD study.

Great thanks to Kersti Peekmaa, Tarmo Rossmann (from Research Administration Office of TTU) and Traugot Läänmäe (from Estonan Patent Agency) for the support and consultations in the field of Intellectual Property in order to formulate the patent claim.

Special thanks:

To journalists Toivo Tänavsuu (Eesti Ekspress) and Kaido Einamaa (Arvutimaailm) for their interest to my inventions and distributing information about them in media, which give me inspiration, strength and motivation to carry on with the research.

To Andres Mellik from Cognuse OÜ [Cognuse] for taking an interest to my invention and arranging a pre-research to determine its business worthiness.

To Professor Mart Tamre for active participation in before mentioned pre-research.

To Enterprise Estonia (EAS) for financing the pre-research.

To The Estonian Information Technology Foundation (EITSA) for the scholarship that allowed me to finish writing the dissertation.

To Research Administration Office of TTU for financing the participations in conferences.

To Archimedes Foundation for financing the participations in conferences.

Finally, I thank my family for the patience and support.

Table of Contents

List of the figures

List of the tables

List of Abbreviations

AIST	Advanced Industrial Science and Technology
bpp	bits per pixel
CHI	Computer Human Interaction (less used than HCI)
CPU	Central Processing Unit
DI	Diffused Illumination
DViT	Digital Vision Touch
dpi	dots per inch
EPA	Estonian Patent Agency
fps	frames per second
FTIR	Frustrated Total Internal Reflection
GUI	Graphical User Interface
GWP	Graphic Window Professional
HCI	Human Computer Interaction (sometimes also referred as CHI)
IADIS	International Association for Development of the Information Society
IEEE	Institute of Electrical and Electronics Engineers
IPL	Image Processing Library
MCCSIS	Multi Conference on Computer Science and Information Systems
MTT	Multi-touch Technology
MTUF	Multi Touch User interface
NUI	Native User interface
RFID	Radio Frequency IDentificator
Taxel	Tactile pixel (tactile surface element)
TFD	Tactile Feedback Device
TUI	Tangible User Interface
TUIO	A Protocol for Table-Top Tangible User Interfaces
UI	User Interface
VECIMS	Virtual Environments, Human-Computer Interfaces and Measurement Systems
VHM	Virtual Height Matrix
Voxel	Volume pixel
VR	Virtual reality
VTA	Virtual Texture Array
WIMP	Windows, Icons, Menus and Pointers

1. Introduction

We use machines,
to create faster and
more powerful machines...

1.1 Background

Information systems have become a natural part of our everyday life. Computers and internet have caused revolutionary changes in science but also in business and entertainment areas. Calculation and data processing tasks that used to take weeks or months to calculate can now be performed in a matter of minutes.

Today we can afford to have such an amount of computing power and information at our fingertips what 50 years ago no one could even dream about. According to Moore's Law the complexity for minimum cost component increases by factor of two per year [Moore 1965] and the number of transistors that can be placed on an integrated circuit will be doubled approximately every two years.

Looking to the computing systems development over last 10-20 years we can say that similar rules apply to the raw computing performance growth and to the speed on which the information becomes available to us through different types of media. Another visionary, Raymond Kurzweil, describes this process as exponentially growing. In his essay, The Law of Accelerating Returns, he extends Moore's Law to describe the ongoing progress of technological growth [Kurzweil 2001].

All this gives us more and more powerful machines capable of handling enormous amounts of information. Unfortunately the interface of computing devices has not gone through the same kind of progress. Already in 1990's it was clear that in development of a new system, more than half of the recourses are spent on the user interface. The 1993 report "New Directions in HCI Education, Research and Practice" states:

"if the interface is ineffective, the system's functionality and usefulness are limited; users become confused, frustrated, and annoyed; developers lose credibility; and the organization is saddled with high support costs and low productivity" [Strong 1995].

In order to study the communication between human and machine, a new subsection appeared in computer science in 1991. That subsection is called Human-Computer Interaction - HCI. Some sources refer it also as Computer-Human Interaction- CHI. The prime goal of HCI is to improve the interaction between the users and computers by making the computers more usable and receptive to the user's needs.

The main directions of HCI research are:

- methodologies and processes for designing interfaces (i.e., given a task and a class of users, design the best possible interface within given constraints, optimizing for a desired property such as learning ability or efficiency of use)
- methods for implementing interfaces (e.g. software toolkits and libraries; efficient algorithms)
- techniques for evaluating and comparing interfaces
- developing new interfaces and interaction techniques
- developing descriptive and predictive models and theories of interaction

A long term goal of HCI is to design systems that minimize the barrier between the human's cognitive model of what they want to accomplish and the computer's understanding of the user's task.

On Figure 1.1 is displayed a visualisation of HCI Topics [Hewett 1992]. As seen, HCI intersects with wide area of other disciplines including engineering, psychology, communications etc.

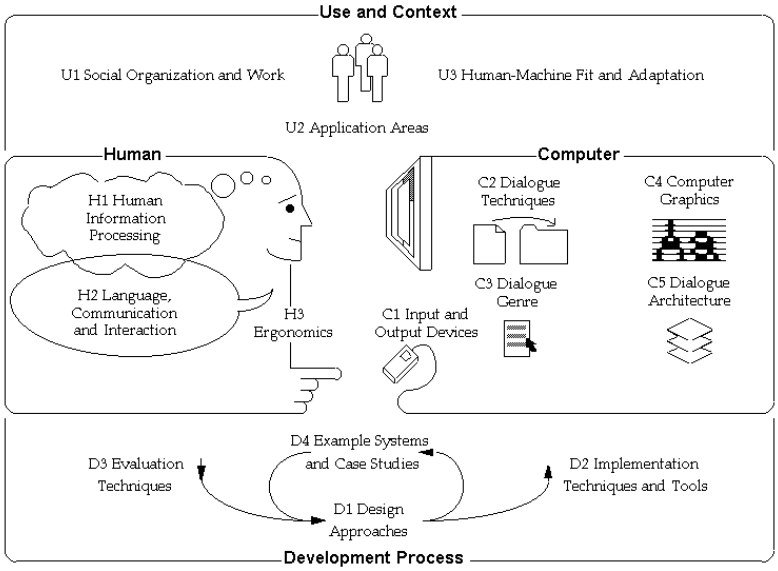


Figure 1.1 HCI topics (picture from Hewett 1992)

Today's most often used interface is Graphical User Interface (GUI) [Johnson 1989] employing WIMP (Windows, Icons, Menus and Pointers). That concept has not changed much since it first appeared (first developed at Xerox PARC 1973, later popularized by the Macintosh computers in 1984) [Van Dam 1997].

The main principles and tools (displayed on figure 1.2) of man-machine communications have been more or less the same for last 30 years. A human user, using input hardware, is doing some predefined actions that are translated to the machine as input data or control information and then used in data processing; the system is then giving a feedback about the system status in the form of visual image or audible signals. The current common tools (hardware) for communication with computers, that utilize mechanical input and visual and/or audio output for feedback, include:

- a display screen (can be touch or even multi-touch sensitive)
- a keyboard
- a mouse and an audio input/output system

Less common options are tablet with stylus, (pseudo) 3D mouse or trackball and audio input systems with voice recognition. The gaming systems have added joystick, gamepad and/or steering wheel and recently also several types of motion sensor / force feedback based controls (Nintendo WII, Sony Playstation 3).

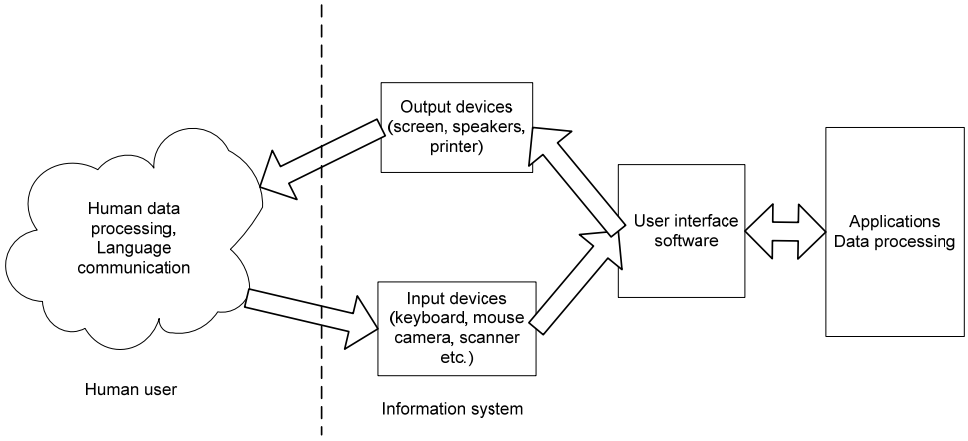


Figure 1.2 Communications between human and machine

1.2 Motivation for this study

With growing amount of information to process and control it is clear that the current UI communication channels, single-point input systems and audio/video based output have become a bottleneck, especially if we also have to consider user groups with some disabilities that limit them using before mentioned communication channels.

This study started out with an objective to create a better method of communication between man and machine. The goal can be achieved using new or previously little used channels: touch, gesture and haptic feedback. From high number of possible research directions this research focuses mainly on how to exploit and improve the methods of creating tactile feedback on fingertips.

This type of interfaces have usually a high price tag due to extensive amount of moving mechanical parts and also because they are produced in relatively small quantities compared to more traditional interface units like display screens and keyboards.

The tactile feedback based user interfaces are also inevitable for people with visual disabilities. Compared with ordinary (keyboard + display screen) user interfaces the Braille display/keyboard systems are ca ten times more expensive [EnableMart UK Ltd] [Sight and Sound Technology Ltd].

That, the motivation of this research is:

- To develop methods that can make tactile feedback providing interfaces technologically simpler, better scalable and affordable, but at the same time keep or improve the accuracy of the tactile feedback.

1.3 Thesis statement

Here is presented a method to implement multi finger input system and supplement it with detailed haptic feedback - individual feedback to each finger which depends on the current finger location, pressure to the screen and also hand orientation on the sensitive surface. The main objective is to improve the dialog channel between the human and computer. So far we had video, audio and manual input tools available. The idea of multi-touch user interface with tactile feedback is to enhance and widen the manual input channel and give to user an immediate haptic feedback.

Haptic feedback devices have been made before, but the main limitation in most of the solutions is scalability. The larger the feedback generating surface, the more raw technology it requires in order to maintain the precision and therefore the cost rises exponentially.

In current work we present solution that can be scaled without major increase in necessary equipment. We also show (in chapter 4) that the same approach and methodology can be used to solve similar problems in slightly different scene, where large amounts of data are presented, but only small areas of it are actually used.

The uniqueness of the approach has been validated by Estonian Patent Agency, when they issued a patent EE05116B1 on 15.12.2008 [Joasoon 2008-3]

1.4 Scope of the research

The research started with an idea to make communication between man and machine more efficient. As the idea of Tactile Feedback Device matured, the decision was made to concentrate the research around that device and accomplish next tasks:

1. Develop a description of the device, its work principles and main components.
2. Get an industry accepted proof (a patent certificate) that the device, its functioning principles and used methods are unique at that moment of time.
3. Develop at least one method to improve some of the enabling technologies needed to operate with the before mentioned device.
4. Find an interested partner, who can benefit from the developed technology and engage a mutually beneficial research project. (Completing this task would prove that the technology is actually useful in real life).

During this thesis we will show that all those tasks were successfully completed: the developed technology was successfully patented, the same methods were applied to improve video processing in optical tracking based multi-touch interface systems and we found an interested partner (Cognuse OÜ), who started a preliminary research project to use this technology in their cognitive retraining application.

1.5 Overview

This thesis is divided into 6 main parts.

Introduction describes the background of the subject, the motivation and the idea. It also clarifies the statement of the thesis.

In Chapter 2. “Related Works” we take a look on technologies and systems that have the same goal or support the technologies developed and described in this thesis. The Related Works chapter is divided into five subcategories (tactile systems, multi-touch systems, gestural systems, interfaces for visually disabled people and computer frameworks) based on main attributes of the specific technology or system.

The main content of the thesis is spread over three chapters (3.-5.), supported by relevant published papers.

Chapter 3. “Haptic Feedback Device” describes device and method patented on 01.2007 under title: “Method and device for haptic duplex communication between computer and user“ and later published on IADIS 2007 conference [Joasoon 2007].

Chapter 4. “Optimizing the positioning process” describes solution to a problem when Multi-touch environment is using camera based tracking system. When a camera with higher resolution is used, the amount of computation power needed to process information feed in real-time grows exponentially [Joasoon 2008-2]. Limited frame tracking provides solution to that problem. The chapter is backed up with a paper „A method to reduce CPU overhead in blob detection and tracking algorithms“ published on IEEE 2008 conference [Joasoon 2008-2].

Chapter 5. ”System usage” describes possible usages of technologies mentioned above. One of those usages was researched together with Jukulab OÜ [Jukulab] and Cognuse OÜ [Cognuse] and then published on VECIMS 2009 [VECIMS 2009] conference in the paper titled: „Employing Haptic Input-Output for Cognitive retraining Applications“ [Joasoon 2009].

Finally the Conclusion wraps together the thesis highlights, the contributions to the field of Information Technology (IT) and the possible direction for future researches that become relevant during this research.

2 Related work

2.1 Introduction

The technologies that we studied and described here are not entirely new. People all around the world are working to create better interfaces. In next subchapters we take a look on some of the more relevant technologies, research projects and product developments. For clarity, they have been divided into separate subcategories:

- **Tactile systems** – Systems that give physically sensible (haptic) feedback.
- **Multi-touch systems** – Interface systems that enable simultaneous virtual object manipulation in several different points.
- **Gestural systems** – Interface systems that allow touch-free virtual object manipulation with gestures.
- **Interfaces for visually disabled people** – Interface systems that are designed to compensate the loss of sight in order to enable those individuals still use Information Technology.
- **Computer frameworks** – software packages and development kits, specially designed for developing the before mentioned interface systems.

2.2 Tactile systems

2.2.1 Haptic Pen: Tactile Feedback Stylus for Touch Screens.

The Haptic Pen (Figure 2.1) is a simple low-cost device that provides individualized tactile feedback and can operate on large touch screens as well as ordinary surfaces. A pen with pressure sensitive tip shaft is combined with a small solenoid to generate a wide range of tactile sensations.

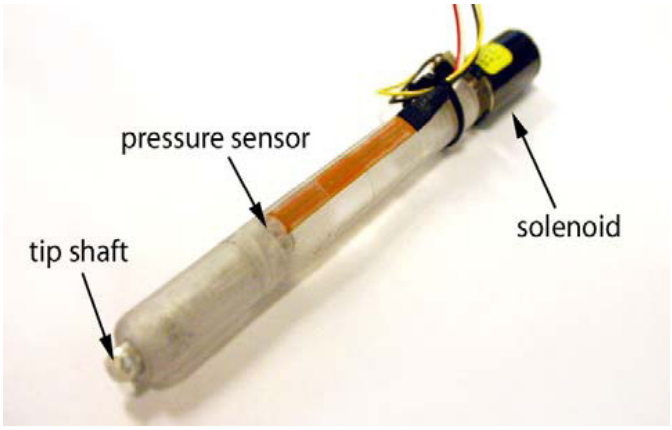


Figure 2.1 Tactile feedback stylus (picture from Lee 2004)

The physical sensations generated by the Haptic pen can be used to enhance our existing interaction with graphical user interfaces as well as to help make modern computing systems more accessible to those with visual or motor impairments [Lee 2004].

2.2.2 Force feedback and texture simulating interface device (Patent Application US 2001043847)

This invention relates to a man-machine interface and in particular to an interface that measures body part positions and provides force, texture, pressure and temperature feedback to a user [Kramer 2001].

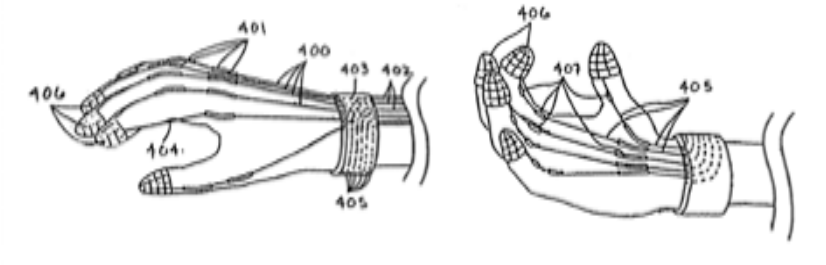


Figure 2.2 Force feedback and texture stimulating device (picture from Kramer 2001)

The patent application describes force feedback control systems, systems that make texture simulation on a selected body part e.g. fingertip (Figure 2.2), systems that are capable of applying pressure to a selected part of a body e. g. hand and also several systems to position and measure hand (or each finger) position/orientation. The described in this thesis finger-mounted pointing devices used in multi-touch user interface with tactile feedback have been developed starting from the designs and methodics, described in this patent application.

2.2.3 SaLT : Small and Lightweight Tactile Display

The SaLT project was published in IEEE International Symposium on Robot and Human Interactive Communication 2008. It is a multi-fingered tactile display module (Figure 2.3) with the spatial resolution of 1,5 mm and temporal resolution of 20Hz.

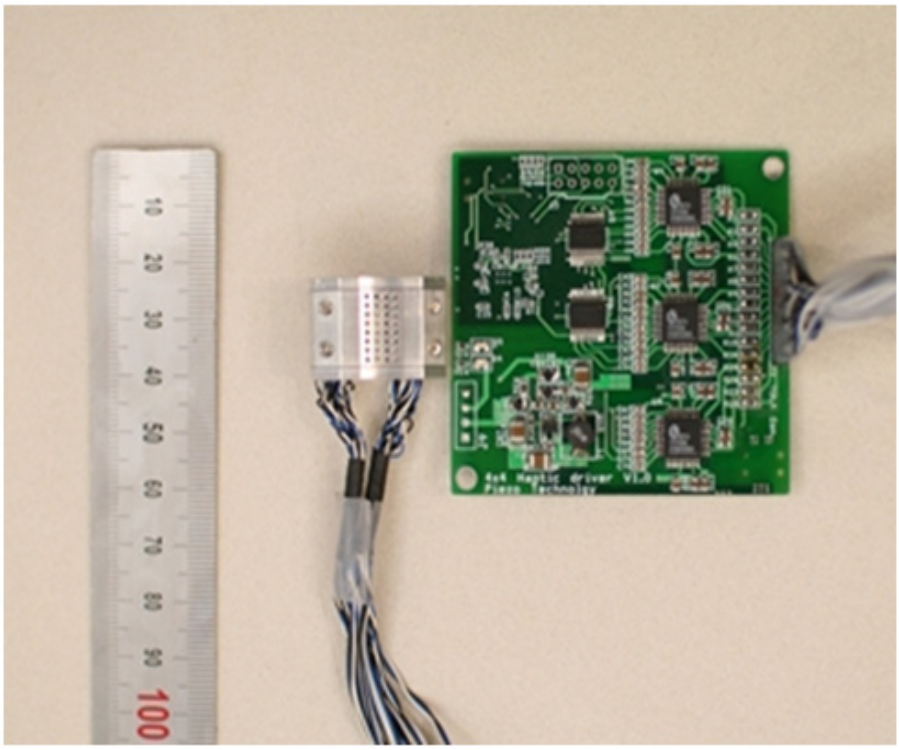


Figure 2.3 SaLT with the electronic driver (picture from robot.kaist.ac.kr)

It is the first available, small and lightweight tactile module that can easily be used in portable solutions (like the tactile feedback device that is discussed later in this thesis.) This system also comprises low-frequency vibration functionality to emulate active touch and therefore make the touch sensation more adequate [Kim 2009].

2.2.4 Senseg

A tactile interface technology from a Finnish company promises "quiet, safe and versatile touch feedback to consumer electronics, including touch screens and gaming applications."[Senseg 2009].

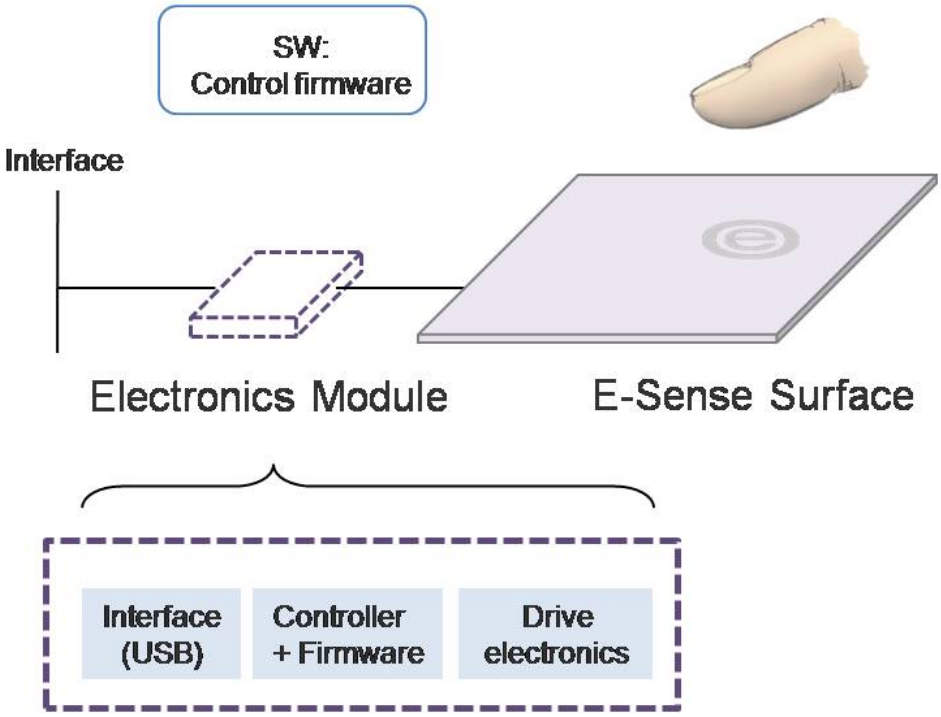


Figure 2.4 Senseg E-Sense™ technology(picture from www.senseg.com)

The tactile feedback is generated by creating a modulated attraction force (using Coulomb force – a force between two or more charged bodies described by Coulomb's law[Elliott 1999]) between finger tissue and E-Sense (name of their technology) surface (Figure 2.4). The company claims that they can accurately control and localise that attraction force in order to create detail vibration-like touch sensations.

2.2.5 Ferro fluid tactile interfaces

Ferro fluid is a liquid that reacts to magnetism. Therefore its surface can be modelled using electromagnetic fields. Based on that effect several tactile interface projects have been created. One example is SnOil [Frey 2009] (Figure 2.5).

SnOil is a 25x25cm basin, filled with ferro fluid that is driven by a grid of 144 electromagnets underneath. The magnets are arranged in identical blocks of 36, which have individual electronics attached. The author claims that it makes the system highly scalable.

The magnetic array enables the creation of 144 independently controlled "fluid-bumps" with maximum elevation of several millimetres.

Figure 2.5 SnOil – a tactile surface based on ferro fluid (picture from www.freymartin.de)

The base material of the ferro fluid is oil which creates a deep black, strong glossy surface. When a "fluid-bump" is created, it not only makes spatial mutation from surrounding surface, but it stands out also visually due to the changing reflection of the surrounding light sources. Therefore it can produce visual images in pixel-graphic as well as pixel typography.

2.3 Multi-touch systems

2.3.1 Apple Inc patent no. 20060097991 – Multipoint touch screen

The invention relates, in one embodiment, to a touch panel (Figure 2.6) having a transparent capacitive sensing medium configured to detect multiple touches or near touches that occur at the same time and at distinct locations in the plane of the touch panel and to produce distinct signals representative of the location of the touches on the plane of the touch panel for each of the multiple touches. This solution is nowadays widely used in Apple iPhones and iPod Touch devices [Hotelling 2006]

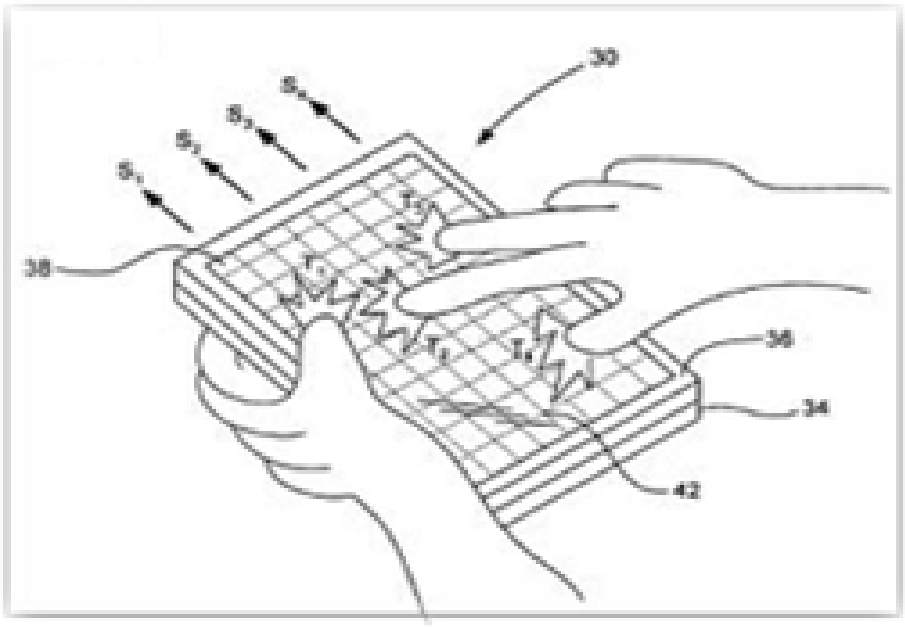


Figure 2.6 Multi-touch sensitive screen (picture from www.freepatentsonline.com)

This invention gives a ground to the presented here device by enabling the registration and positioning of multiple simultaneous touches to the surface. This is essential in order to widen the communication channel, compared to the „single touch" interfaces that are realized in traditional touch sensitive or mouse controlled solutions.

2.3.2 Multi-touch interaction using FTIR effect

This research project introduced an inexpensive and simple method called Frustrated Total Internal Reflection [Han 2005] (FTIR, a technique that has been widely used in biometrics for fingerprint image acquisition) to enable multi-touch sensing.

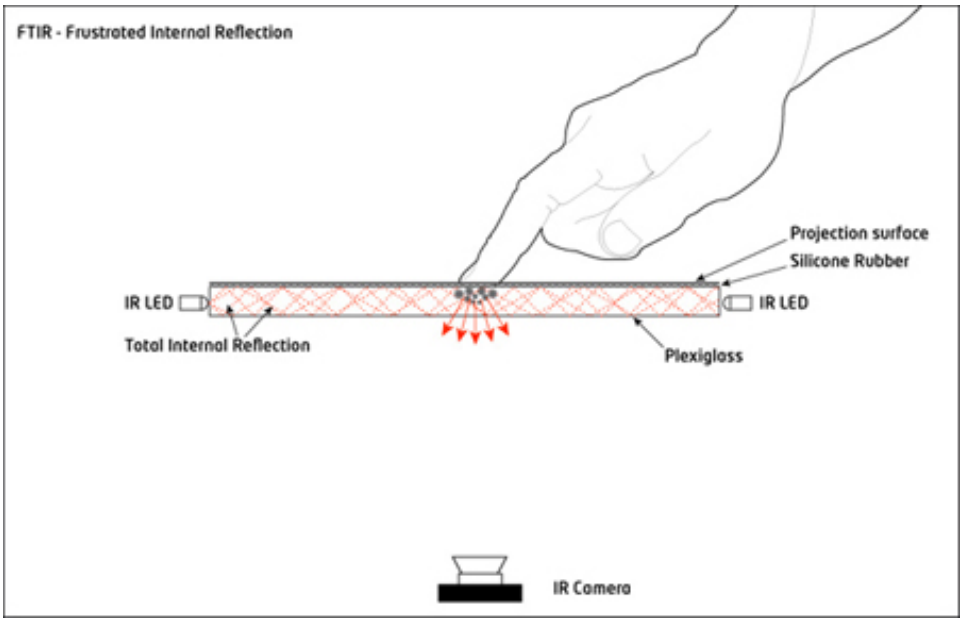


Figure 2.7 FTIR method principle (picture from wiki.nuigroup.com)

The principle of FTIR is shown on figure 2.7. Total Internal Reflection describes a condition present in certain materials when light enters one material from another material with a higher refractive index, at an angle of incidence greater than a specific angle [Gettys 1989].

When creating a multi-touch interface, using this effect, the inside of an acrylic panel (touch surface) is flooded with infrared light. That light is trapped inside the panel because of Total Internal Reflection principle[Zhu 1986]. When an object (user's fingertip) is touching the surface of that acrylic panel, the light rays will be *frustrated*, since they can now pass through into the contact material. The result is a bright "blob" of reflecting light on the other side of the surface. Those blobs will be registered by IR camera, which is located some distance away under the touch surface.

For better results a silicone rubber layer is often used as a compliant surface on top of the acrylic. It increases the touch sensitivity and helps making dragging smoother as getting the blobs visible needs very little or no pressure. In opposite when using bare acrylic, user needs to press hard or have oily fingers in order to set off the FTIR effect.

The working prototype (Figure 2.8), introduced in 2006, consists 36“x27“ rear projection drafting table style implementation that had a sensing resolution better than 0.1“ at 50Hz [Han 2005].

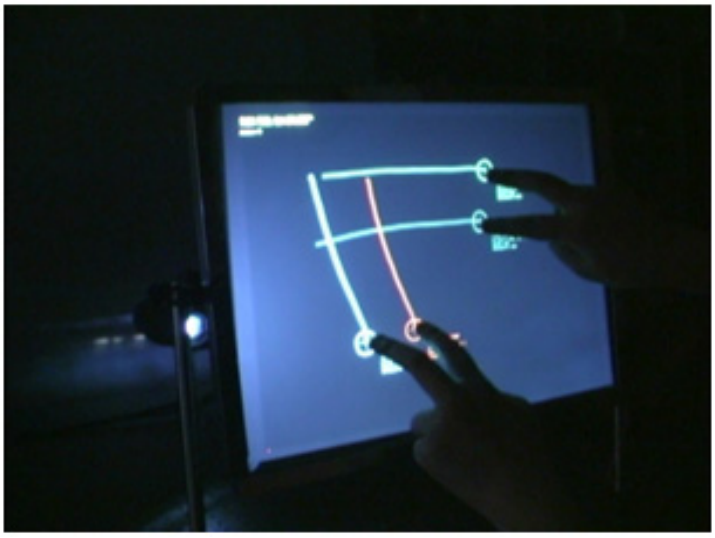


Figure 2.8 Multi-touch screen implementation, using FTIR technology(picture from cs.nyu.edu/~jhan/)

This technology has been further developed and commercialized by a company, founded by Jeff Han and called Perceptive Pixel (www.perceptivepixel.com). The solution is called Multi-Touch Wall and it is 81”x48” (205cm x 121cm) screen that accommodates up to four users, who can all work simultaneously in a collaborative setting[Perceptive Pixel].

2.3.2 Multi-touch interaction using Rear Diffused Illumination effect

Rear Diffused Illumination (DI) is another effect that allows affordable implementations of multi-touch interface system. Its working principles, illustrated on Figure 2.9, are as follows:

Infrared light is directed to the screen from illuminators (Infrared LED’s) that are located below the (transparent material like glass, acrylic etc.) touch surface. A diffuser (vellum, Mylar, Lee Filter etc.) is attached on top or underneath the touch surface. When an object touches the surface, it reflects IR light back and that light

is then registered by the IR camera that is located at some distance under the surface.

Depending on the diffuser, this method can be used to detect only touches, but also objects that are hovering over the surface.

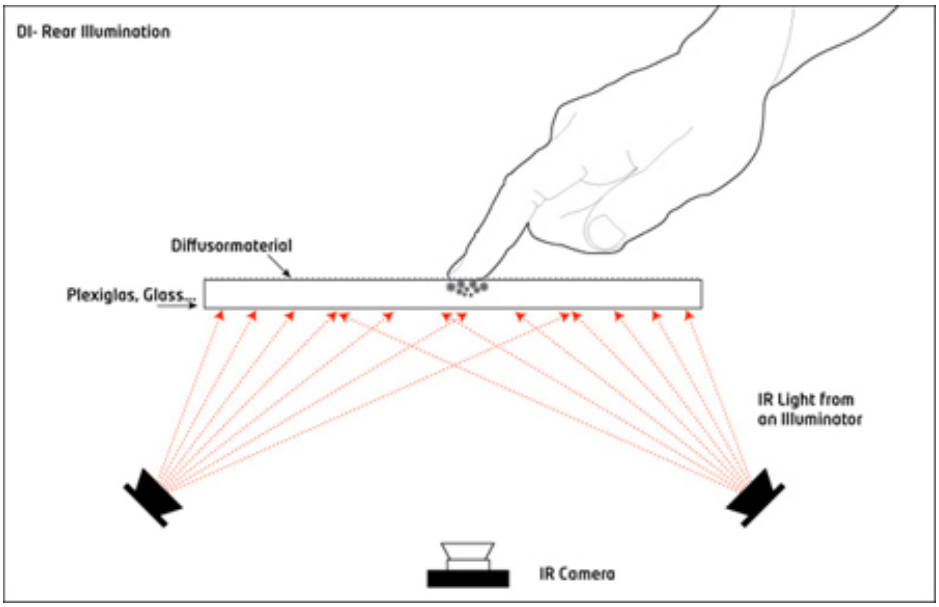


Figure 2.9 Rear DI principle (picture from wiki.nuigroup.com)

In commercially available products, RearDI is used in Microsoft Surface.

Microsoft Surface with manufacturer codename Milan is a multi-touch product, combining software and hardware that allows a user (or multiple users, up to 52 simultaneous touches [Eden 2003] to manipulate digital content on screen using gesture recognition It is based on multiple infrared cameras and can recognize not only finger touches, but also tagged objects like phones and digital cameras in order to interact with them.

MS Surface (Figure 2.10) is a 30-inch (76 cm) display in a table-like form factor. The Surface tabletop is acrylic with a XGA DLP projector underneath the surface which projects an image onto its underside, while four cameras inside the housing record reflections of infrared light from objects and human fingertips on the surface [Microsoft Surface][Vale 2007].

Figure 2.10 Microsoft surface (picture from www.microsoft.com/surface/)

2.3.3 DViT technology

Digital Vision Touch (DViT) technology was developed by SMART Technologies. This technology does not require touch screen nor special pointers and is based on four digital cameras (Figure 2.11), one in each corner of the screen. The touch point is calculated by triangulation while using data from two of the best suitable cameras. Due to the technology restrictions it is able to register maximum of two simultaneous touching fingers.

Figure 2.11 DViT implementation with camera in the corner(picture from www2.smarttech.com)

System has the ability to detect two statuses of the hand: direct contact with the screen and a status when the hand is just hovering above the screen.

The main benefit of this solution is that it can be applied on current (not touch sensitive) system without the need of modifying the existing one [Smart 2003].

2.4 Interfaces for blind

2.4.1 Tactile Mouse

Tactile mouse was mainly developed for the purpose of providing for blind people the access to graphical information. In principle, it is an ordinary computer mouse (Figure 2.12) with a small tactile display attached to it. Combination of the mouse movement and tactile feedback creates the feeling like if the user is exploring a tangible drawing.

Figure 2.12 Tactile mouse(picture from www.nise.go.jp)

The display unit consists of 32 movable pins that are moving up and down in order to create the tactile image of what is on the screen on mouse pointer's position [T. Ishida 2000],[T. Watanabe 2000], [T. Watanabe 1998].

2.4.2 Braille display

Classically, the Braille display is an electro-mechanical tactile device that consists of a row of special cells that allow displaying Braille characters [AFB]. Each cell has 6 or 8 pins, made from nylon or metal. They are controlled electronically to move up and down and raise dots through holes in a flat surface. As the little pins of each cell pop up and down they form a line of Braille text that can be read by touch.

Figure 2.13 Braille Star 80 (picture from www.handytech.us)

The mechanism is usually based on piezo effect or solenoids. Most commercial models (like Handytech Braille Star on Figure 2.13) employ 40-80 cells in a single row [Handytech].

2.4.3 Graphic Window Professional (GWP)

GWP (Figure 2.14) is a device, developed by Handy Tech Elektronik GmbH that allows visually disabled people to touch the graphical information on the Windows desktop.

The graphical images are converted into tactile pictures of 24x16 taxels by software layer developed by Handitec. The observation frame is moved around the screen using arrow keys. The system also employs a zoom functionality that enables image enlargements up to 1:1 (single pixel on the screen is represented by single tactile pin on the tactile display)

Figure 2.14 Graphic Window Professional (picture from www.handytech.us)

2.5 Gesture based systems

2.5.1 Multi-finger gestural interaction with 3D volumetric displays

Volumetric displays produce volume filling 3D images. The images are made up from volume elements (Voxels – volume pixels), that emit visible light from the location in which they appear[Favalora 2005]. This technology that presents 3D visualization and interaction requires also new type of interface, which will allow natural 3D manipulation of objects. The techniques, described in this research enable manipulation from any viewpoint (360^{o}) around the display (Figure 2.15).

Figure 2.15 Volumetric display with camera based finger tracking system (picture from www.acm.org)

The input system is based on Vicon motion tracking system [Vicon] to position and track markers attached to the user's fingers [Grossman 2004].

2.5.2 Various Nintendo WII remote based projects

Currently one of the most sophisticated input devices for game consoles and computers is Nintendo Wii remote (Figure 2.16). According to Nintendo 2008 annual report as of 31 March 2008, Nintendo has sold 24,5 million Wii game consoles worldwide [Nintendo 2008]. This fact makes the Wii Remote one of the most common computer interface device. It contains:

- 1024x768 infrared camera with built in blob detection and tracking of up to 4 points.
- +/-3g / 8bit 3-axis accelerometer
- Expansion port
- Bluetooth communication with main module

Figure 2.16 Nintendo WII remote (picture from www.everyjoe.com)

Jonny Chung Lee shows in his Wii remote projects [Lee 2008-2], that it is possible to create a cheap and simple 3D multi-touch interface by using a $40 Wii Remote controller and some really cheap accessories like pieces of reflective tape etc [Lee 2008-1].

2.5.3 g-Speak Spatial Operating Environment

g-Speak, the spatial operating environment (Figure 2.17), is a system developed by Oblong Industries.

Figure 2.17 g-Speak – the Minority Report style interface (picture from oblong.com)

It is a combination of gestural input output interface, recombinant networking (name of Oblong technology), that supplies structured coordination with data

interchange, and Real-world Pixels (an Oblong term to describe their technological solution to output information to multiple screens) [Oblong]

2.5.4 Mgestyk

Mgestyk (Figure 2.18) is a solution developed by Mgestyk Technologies (Canada) that uses a 3D camera to capture hand gestures.

Figure 2.18 Mgestyk - gesture based user interface (picture from www.mgestyk.com)

It uses 3D data to process the gestures (without the need of wearing special gloves or any markers for fingers like most of above mentioned solutions), and translate them into control information that is suitable for Windows-based games or applications. At the time this section is written (summer 2009) the system is still not commercially available [Mgestyk].

2.6 Computer frameworks

In this section we describe some computer frameworks that have been or can be used (in future works) in research and development of current thesis material.

2.6.1 reacTIVision

reacTIVision (Figure 2.19) is an open source, cross-platform computer vision framework, developed by Martin Kaltenbrunner and Ross Bencina at the Music Technology Group at the Universitat Pompeu Fabra in Barcelona.

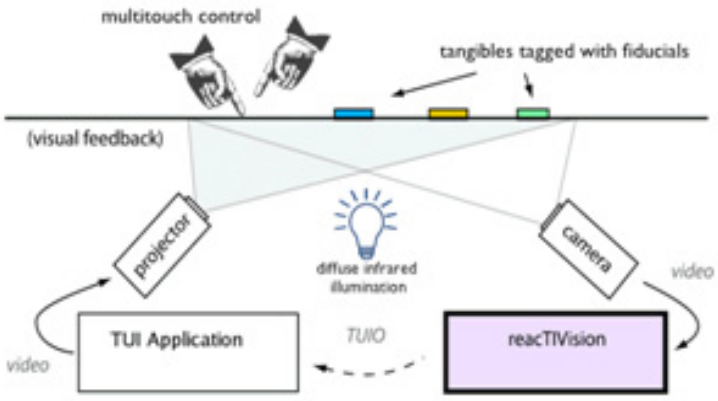


Figure 2.19 reacTIVision framework diagram (picture from reactivision.sourceforge.net)

The idea behind reacTIVision is fast and robust tracking of fiducial markers (Figure 2.20) attached onto physical objects, as well as for multi-touch finger tracking systems [Kaltenbrunner 2008].

The main motivation behind designing reacTIVision platform was to create a toolkit for rapid development of table based tangible user interfaces (TUI) and multi-touch interactive surfaces [Microsoft Surface].

reacTIVision is a stand-alone application, which sends Open Sound Control [Wright 2005] messages via UDP port 3333 to any connected client application. It implements the TUIO protocol [Kaltenbrunner 2005], which was specially designed within this project for transmitting the state of tangible objects and multi-touch events on a table surface.

Figure 2.20 Fiducial markers (picture from reactivision.sourceforge.net)

2.6.2 OpenCV

OpenCV is an open source computer vision library originally developed by Intel under name IPL (image processing library) [Bradski 2008]. It is a cross-platform library that can run on Mac OS X, Windows and Linux and supports most versions of C/C++ (including but not limited to MS Visual C++ and Borland/Codegear C++ builder). Currently it is developed and supported by several online communities [Stavens 2008] [Bradski 2008]

2.6.3 cvBlobsLib

cvBlobsLib is a library developed by OpenCV community member and it provides functions to extract blobs from images, but also to manipulate, filter and extract results from the extracted blobs. Works only with greyscale images [Blob Extraction Library].

3 Haptic Feedback Device

„Tactile graphics are images that uses raised surfaces so that a visually impaired person can feel them. They are used to convey non-textual information such as maps, paintings, graphs, diagrams, etc."

[NationMaster Encyclopaedia]

This chapter describes the main contribution of this thesis - multi finger input system that is enhanced with individual haptic feedback to each finger. The content of that feedback is determined by each fingers location, pressure to the screen and orientation on the surface. Here we present first some background information, the idea and motivation and then describe the hardware and methods to operate that hardware. In the summary of this chapter we discuss future development directions, some possible usages and advantages/disadvantages of the device. The usability is described more detail in the chapter „System usage" under „Use in cognitive learning process" subsection.

3.1 Introduction

So far we had audio/video for output and (single point) manual input tools available. The main objective of our study is to enhance and widen the human-computer dialog channel by allowing multi finger input functionality with an immediate and detail individual haptic feedback to each touching finger.

As this device is the cornerstone of the thesis and the research project, in order ensure the uniqueness of the device it was patented (claim issued in Jan 2007, patent registered on Dec. 2008) [Joasoon 2008-3] The description and work principles of the Haptic Feedback device were also published on IADIS MCMSI conference [Joasoon 2007]

3.2 Tactile feedback and haptic surfaces

In this section we describe the essence of the haptic surface, the ways to describe it mathematically and also consider the question „how to digitize it?".

Tactile feedback is based on touch sensation – the sensation produced by the pressure receptors of the skin (Figure 3.1). Touch receptors are types of specialized nerve endings. Meissner's corpuscles detect fine touch and are found in hairless parts of the body, such as the lips, palms, and fingertip; other types of receptor are sensitive to pressure, stretching of the skin, vibration, or hair movements. In this work we concentrate on the touch sensations on fingertips.

The human haptic perception consists from two distinct sensing components:

- Tactile or Cutaneous – Tactile (or cutaneous) receptors are located in the dermis and epidermis, and convey information about contact pressure distribution, indentation, temperature, etc.
- Kinaesthetic – Kinaesthetic receptors are mostly located in tendons, muscles, and particular joints. Kinaesthesia refers to sensations related to gross force/motion of the fingers or of the hand: for the information it conveys, it is comparable to what a person would feel wearing thick gloves or thimbles on his/her hand [Fundamentals of haptic].

The Haptic Feedback Device that we discuss in this thesis provides only cutaneous information

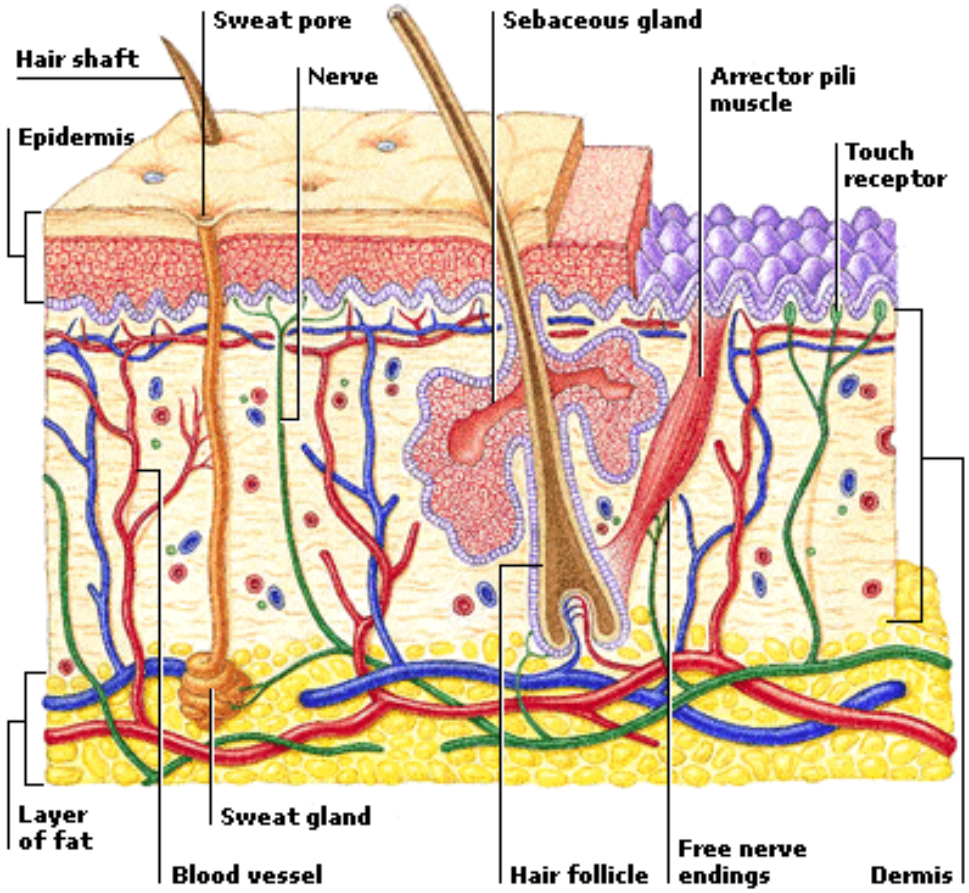


Figure 3.1 Skin structure (picture from www.factmonster.com)

When talking about Haptic surface (Figure 3.2) we mean a plane with raised surface texture that can be sensed in a form of touching (tactile graphics). And the

images that are displayed in tactile graphics can change over the time. This surface can be described as a function:

$$T_h = F(x, y, t) \tag{3.1}$$

Here T_h is the height of the surface texture point in coordinates *x,y* at time *t*.

The function 3.1 describes a tactile surface with unlimited size and over unlimited period of time. In reality there are limits on the size of the surface, the maximum height (and depth) and also it is used over a certain period of time.

$$T_{h\,min} \leq F(x, y, t) \leq T_{h\,max} \tag{3.2}$$

$$X_{min} \leq x \leq X_{max} \tag{3.3}$$

$$Y_{min} \leq y \leq Y_{max} \tag{3.4}$$

$$T_{min} \leq t \leq T_{max} \tag{3.5}$$

Thus T_h is a bounded three dimensional functional with bounded independent variables, which is continuous over its domain of definition (no „gaps“ or undefined values on the area).

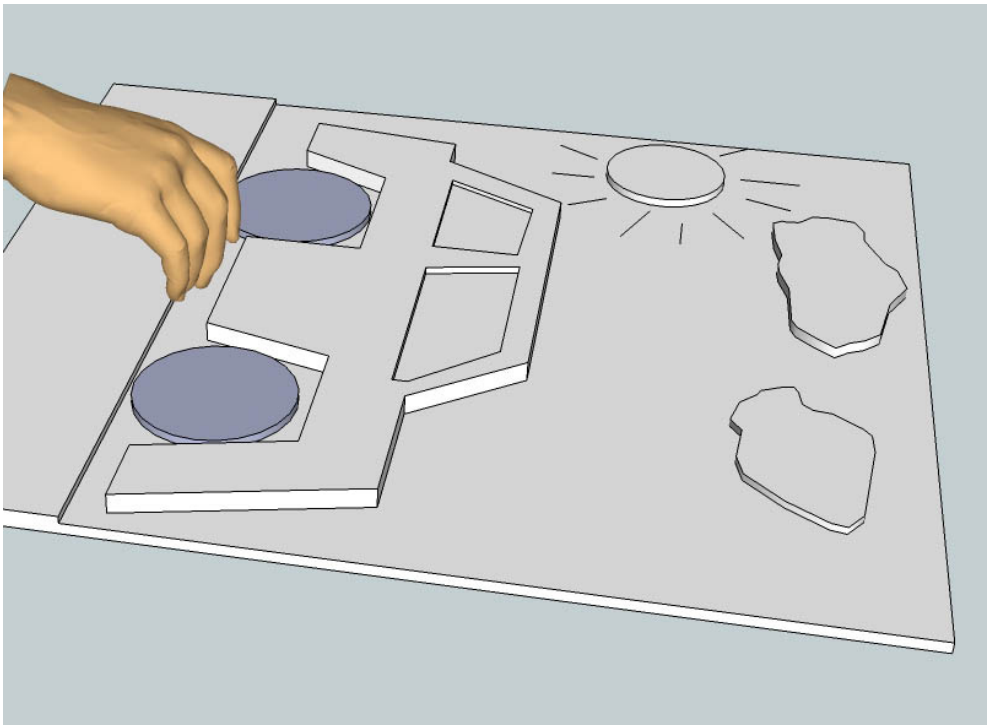


Figure 3.2 Tactile surface

If we want this to be digitized, we need to determine the distance between digitized points that will be small enough to make the skin feel like the continuous surface and not consisting of separate elements (taxels).

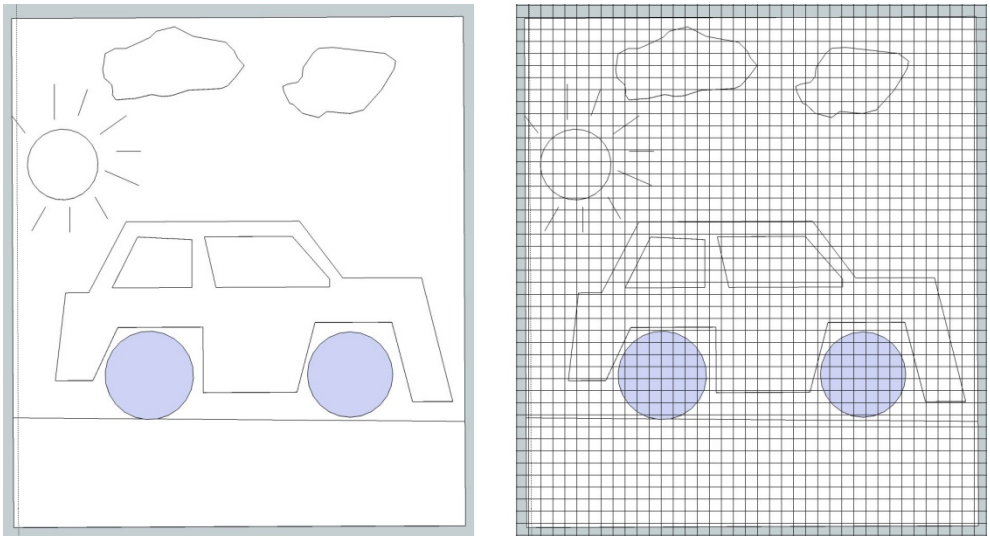


Figure 3.3 Tactile surface with digitalization mask

The methodics to determine those parameters will be looked more closely in chapter 3.3.3, where the design of haptic feedback providing matrix element is discussed.

Once the parameters have been determined, the process of digitalization divides into two parts:

- Sampling – means digitizing the coordinate values (x, y and t).
- Quantization – means digitalization of the height values (how much is that point raised above the "ground zero").

The result of digitalization is a matrix of real numbers. Let's assume that the tactile surface $f(x, y)$ is sampled so that the resulting digital image of that surface has M rows and N columns. The values of the coordinates (x, y) become discrete quantities. The values at the origin are $(x, y) = (0,0)$ an the values at the end are $(x, y) = (M - 1, N - 1)$. Important to notice is that those values are not the physical coordinates of where the surface was sampled.

Now we can write the complete $M\ x\ N$ tactile surface in the following form as a matrix:

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0,N-1) \\ f(1,0) & f(1,1) & \dots & f(1,N-1) \\ \vdots & \vdots & \ddots & \vdots \\ f(M-1,0) & f(M-1,1) & \dots & f(M-1,N-1) \end{bmatrix} \tag{3.6}$$

The right part of this formula is by definition a digital tactile surface. Each element of the matrix array is a tactile surface element or a taxel.

3.2 Idea

Different kind of tactile display devices that convert graphical information into tactile pattern and display it on a touch panel have been around for some time now. For instance the device on figure 17, developed by the Institute for Human Science and Biomedical Engineering of the National Institute of Advanced Industrial Science and Technology, has 1536 up and down moving pins arranged at 2.4-mm spacing, 32 X 48 [AIST].

Most of them however have one disadvantage. In order to display adequate surface texture and relief information, the centres of concurrent display pins have to be no more than 2,5 mm away from each other [Kenshalo 1968]. Building a large scale screen is not only expensive, but it also provides sufficient technological challenge.

In order to build a 19" tactile display screen, that provides seamlessly continuous touchable texture information, more than 15000 actively moving pins are needed. Placing that amount of mechanically moving devices onto relatively small area and then controlling them all individually makes the system not only extremely complex and expensive to build but also creates other issues, like power distribution and cooling.

Building a touchable analogue of computer visual screen is also illogical – human touch has much more limited range than vision. Humans can perceive only this part of tactile information what is directly under their fingertips, all other pins them are useless.

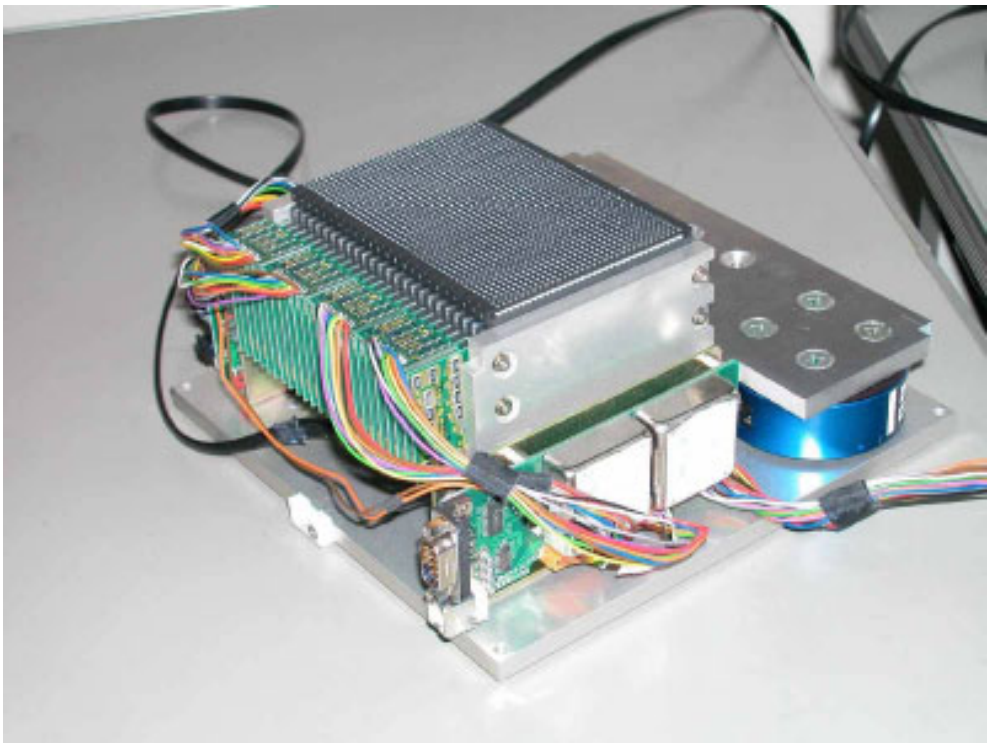


Figure 3.3 The prototype tactile display (picture from www.aist.go.jp)

If the touch area (and control of the objects on that screen) is merely limited to (all 10) fingertips, then only about 6,5% of the pins are in active use at every moment, all other pins only overburden the device's communication channel. Minimizing the communication only to the pins that are directly under the fingertip(s) and moving them around on the screen as the finger(s) move allows to decrease the information flow in device's communication channel by several oders.

This solution brings in several technical difficulties, like to identify which finger is in which location and the fingers orientation.

To address this issue we have implemented dual positioning system that is able to identify and also capable to calculate the orientation of the fingers. Those techniques will be more closely addressed in next two sections „The Hardware" and „The Method".

3.3 The Hardware

In this section we are discussing the physical components of the device: what components does the system need, why and what specifications do those components have to meet.

The tasks that the hardware has to complete are following:

- Display the visual image
- Register accurately all touching points on the display surface
- Position all fingers (that carry the thimble styluses) locations and identify them over the display surface.
- Position the location of the centre of palm or back of hand (this will be needed to calculate the orientation of each finger toward the display)
- Measure the force the finger is using to press the stylus against the display surface.
- Provide adequate, individual and correctly oriented haptic feedback to every finger (that is carrying the thimble stylus).

In order to fulfil these functionality requirements the hardware part of the solution consists of: multi-touch sensitive panel (Figure 3.4 position 1.), finger mountable pointing devices - thimble styluses (Figure 3.4 position 2) and secondary positioning hardware (Figure 3.4 position 3) to determine the ID and orientation of each finger using radio locators, that triangulate the RFID elements in thimble styluses.

There is also a control unit (Figure 3.4 position 4) that handles data communication between each connected thimble stylus and the computer. The preferred way of communication is wireless (Bluetooth or some other short range data communication protocol.) The control unit also houses power cells and a RFID element that can be identified and positioned by the radio locators.

3.3.1 Multi-touch panel

In order to decide what tactile feedback information to send to each finger the positions of the touching fingers has to be determined with sufficient accuracy. For this is used a two level approach.

First step is positioning via physical touch. It gives good accuracy for each touch point, but does not identify which finger made the touch. This is a problem of current technology as the attempts to implement identification functionality here (fiducial markers for example) immediately reduces the positioning accuracy.

For ordinary people the multi-touch panel would be a panel with visual output capabilities (like Apple Multi-touch panel or Microsoft Surface).

If the interface is meant to be used by people with visual disabilities, the functionality of displaying visual information becomes secondary – the main

requirement is that it has to register multiple simultaneous touches and send their coordinates to the computer.

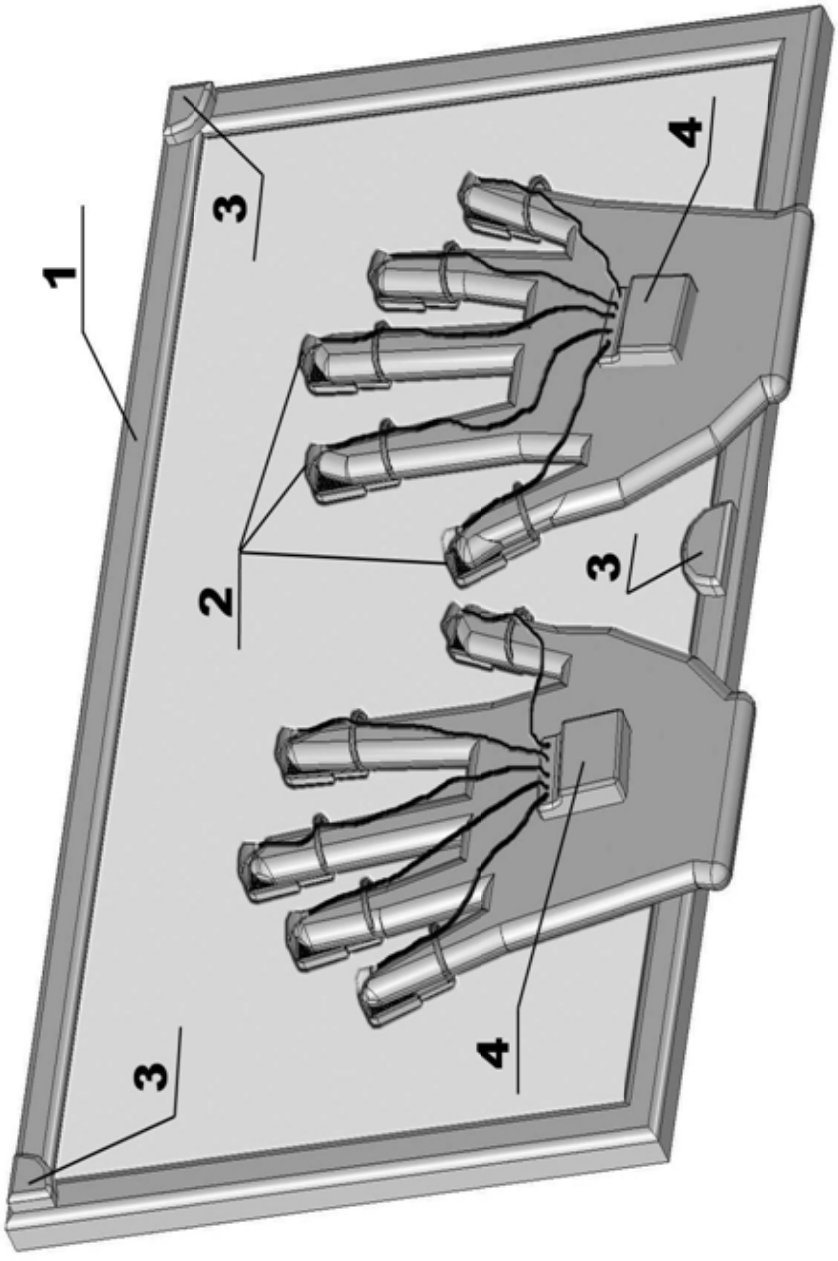


Figure 3.4 Overview of the haptic device with multi-touch screen

The most used technology today to capture multi-touch information is FTIR (used also in Microsoft Surface) that employs a camera to capture touch points and blob-detection software to process the video stream frame-by-frame and extract the coordinates. If the panel is larger, then also the camera resolution has to be higher to maintain touch detection accuracy. But when the camera resolution becomes higher, the amount of information in the processed video stream is also larger and to process it (blob-detection) consumes significant amount of CPU power. This problem is also addressed in this research and one possible solution is offered in the Chapter 4 „Optimizing the positioning process".

3.3.2 Thimble styluses

Each thimble stylus (Figure 3.5) contains a matrix element providing haptic feedback (Figure 3.5 pos. 3) stylus-like tip (Figure 3.5 pos. 2) that measures the amount of pressure that the finger is applying to the surface and a RFID element for positioning and identification. The positioning and identification process is performed by radio locators using triangulation and trilateration; its purpose is to determine the location and orientation of each finger. The reason for having two positioning mechanisms is explained in section „The Method".

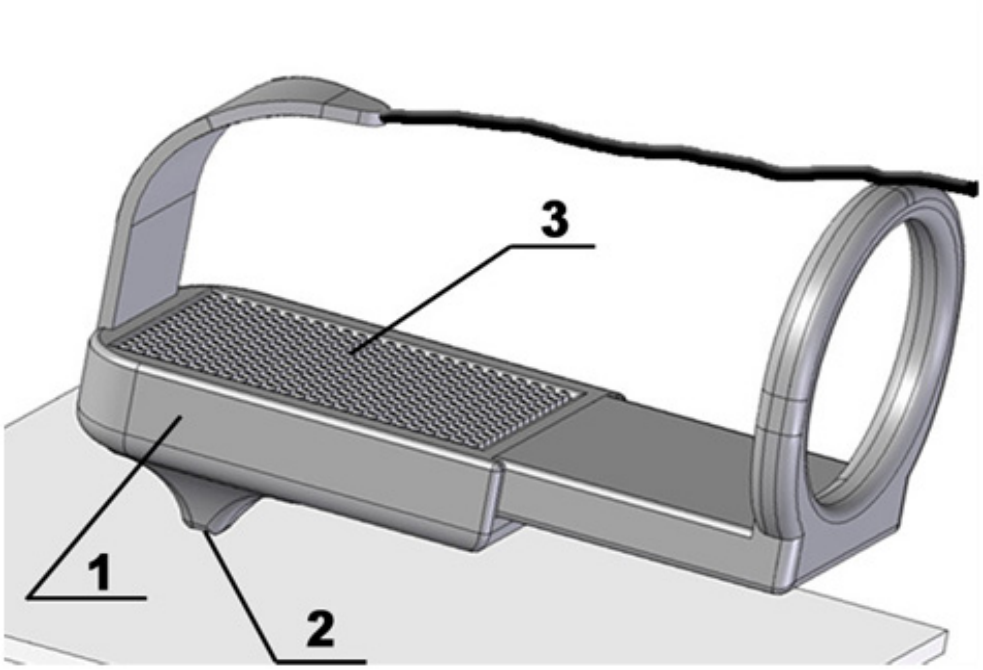


Figure 3.5 Finger mounted pointing device- thimble stylus

3.3.3 Haptic feedback providing element

The haptic feedback providing matrix element is one of the key components of this device. It has to comply with very specific requirements in order to be able to provide an adequate and continuous haptic sensation. To understand those exact requirements we had to research the mechanisms of how fingertip receives the feel of haptic sensation (that was described in p3.2).

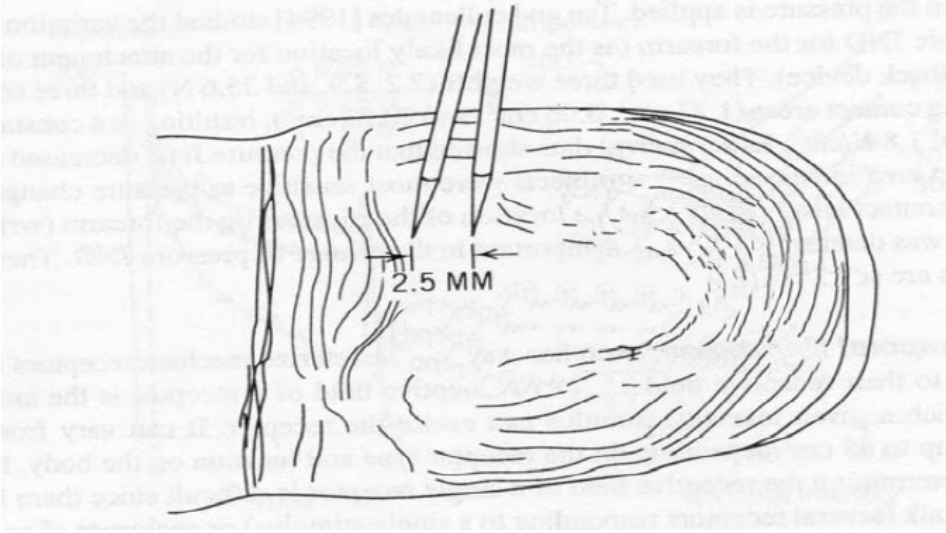


Figure 3.6 Two point Limen test on the fingertip [Burdea 2003]

In order to create sensations that feel realistic, the system resolution (spatial and timing) must be as good as the resolution of the sensing organ (in this case fingertip).

The ordinary two point Limen test [Burdea 2003] indicates that the resolution should be less than 2,5mm between the centres of two tactile elements in haptic matrix.

The main haptic parameters of fingertip are presented in Table 3.1

It is quite clear that a haptic feedback matrix with elements whose vertical movement can be controlled with 0.06 micron accuracy is unreasonably expensive and complicated, but the two other parameters (minimal reaction time and spatial resolution) are quite achievable [Kim 2009]. Based on those initial requirements we can easily determine that the haptic feedback matrix should be 15 X 25mm size and contain 77 movable tactile pins that are placed 2,5mm apart (from centre to centre), figure 3.7.

Table 3.1 The limits of haptic sensitivity of human fingertip [LaMotte 1991]

Haptic parameter	Minimal sensible value
Minimal reaction time / max distinguishable pulsation	5-20 ms (50-200Hz)
Minimal height of detectable texture	0.06 microns.
Density of receptors /spatial resolution	~2,3 mm

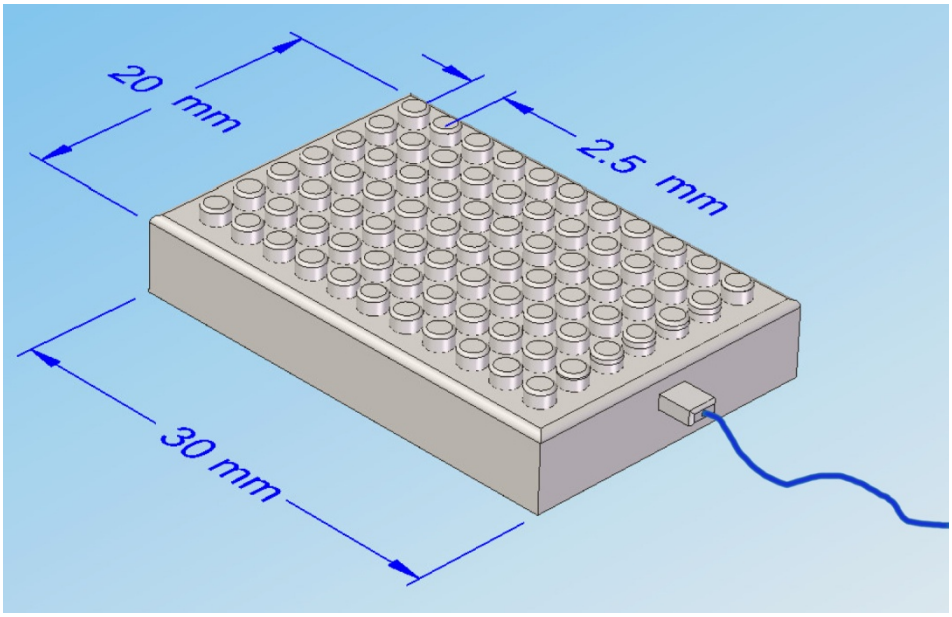


Figure 3.7 Tactile feedback element (sketch, not an actual prototype model)

3.3.4 Control unit

The control unit (Figure 3.4 position 4) is fixed to users back of the hand. Its main tasks are:

- Maintain two-ways data communication with the computer (exact dataflow and functionality will be described in section „The Method"). The preferred mean for it is Bluetooth, but other options like Wi-Fi, RF or even cable with USB connection (in this case no additional power cells are required) are possible.
- Maintain two ways data communication with thimble styluses, control the haptic feedback providing matrixes and receive pressure information from sensors that are located in each thimble stylus pointing tip.
- Control unit has also RFID tag that is positioned by the secondary positioning hardware and provides the system with coordinates of „the centre of the hand" that are needed to calculate the orientation of each finger.
- If it is a wireless solution, the control unit will be the place where power cell unit are located.

3.3.5 Secondary positioning hardware

Secondary positioning hardware consists of three radio locators (Figure 3.4. position 3) placed on the corners (and edges) of the multi-touch display screen and RFID tags placed in thimble styluses (Figure 3.4 position 2) and control units (Figure 3.4 position 4).

The positioning works using RFID positioning [Song 2007] and in return provides the coordinates of each thimble stylus and control unit. This information is later used to determine where is every finger and by calculating the finger and relevant hand control unit coordinates, to determine what is the orientation of each finger against the multi-touch screen.

3.4 The Method

In this section we will describe the logical layer of the device i.e. how the different hardware parts act together, how and what information is sent from one unit to another and how the whole process is going to help to fulfil the main task of this system.

The device works as follows (Figure 3.8 and Figure 3.9). The user mounts the thimble styluses to his/her fingers and control unit(s) to the back of the hand(s) and touches the screen surface. As mentioned earlier this surface can be a multi-touch sensitive display or just a multi-touch sensitive tablet. The surface will send the touch coordinates to the application software that runs on the computer. This first level of positioning gives good accuracy (<0,1" according to Perceptive Pixels product sheet [Han 2005]) of touch coordinates, but no information about which finger causes which touch. This is because today's technology of (multi) touch screens lack this type of functionality. Therefore a second level positioning is implemented.

Each finger-mounted pointing device will then be positioned and identified with the help of the special hardware. In addition to that, there are positionable beacons in the control units which are strapped to the back of the user's hands and which will also be positioned and identified. This second level positioning gives less accurate coordinates, but identifies each finger – that is necessary for calculating the orientation of each finger and to be able to send each finger individualised haptic feedback information. The calculation is performed, using the coordinates of fingertip and the coordinates of the control unit strapped to the back of the same hand. This information is also sent to the application software. From the received information, application software will determine and calculate the location and orientation of every finger.

Application software creates together with the visual image array (video image in video card's memory) also a virtual texture array. Visual image array represents all the pixels on the output display screen. Virtual texture array has the same amount (and configuration) of elements as the visual image array, but every element represents the height of the current point on the screen.

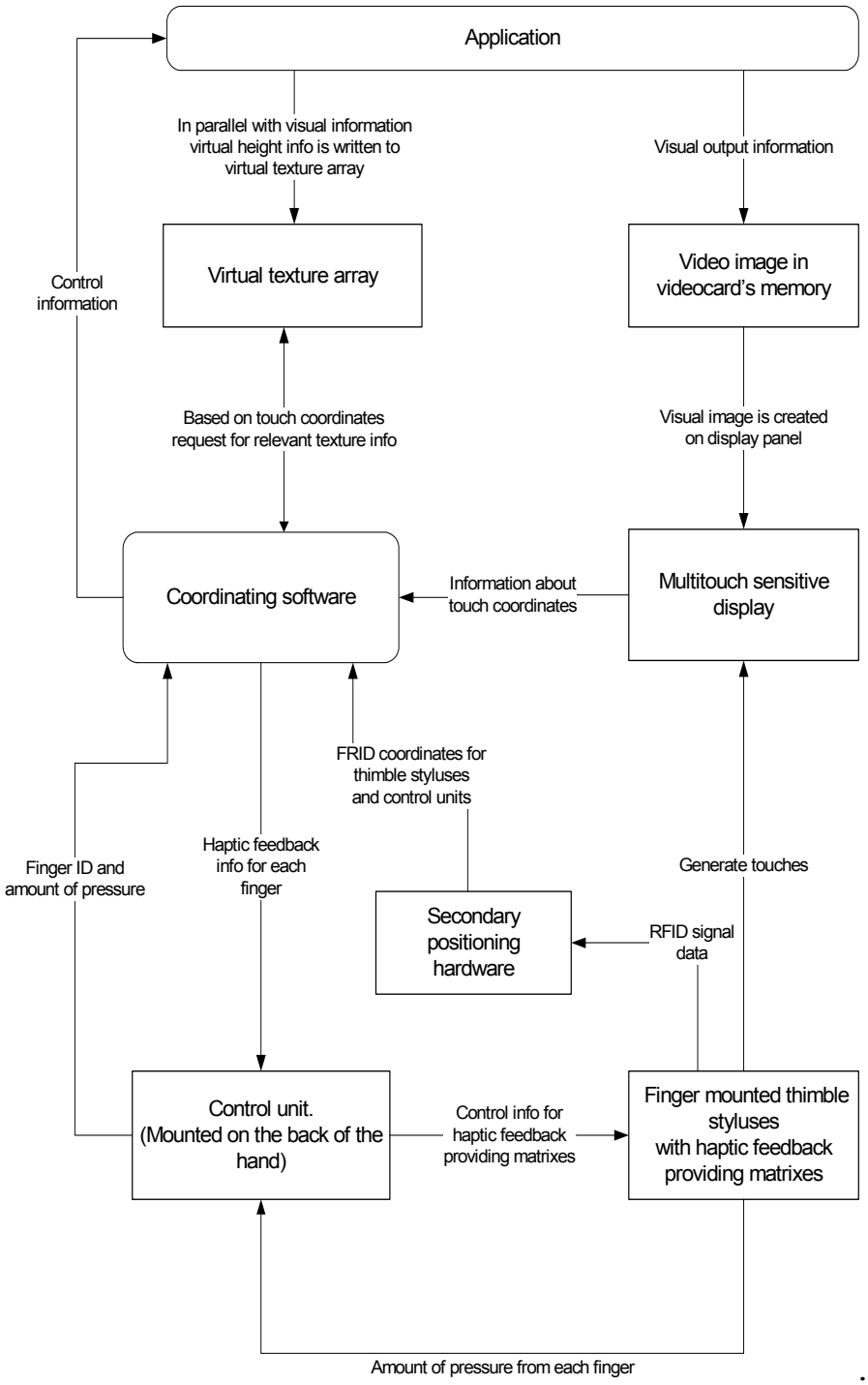


Figure 3.8 System overview

Knowing the touch point and orientation of the finger, the application will retrieve the necessary texture information from the visual texture array, rotate it, if necessary, to match the orientation of the finger and send this info to the control block, which will then generate a haptic sensation for appropriate fingertip using the matrix element (Figure 3.5 pos. 3) located in the finger mounted pointing device.

In parallel to the process described above, the multi-touch information which consists of touch locations and the amount of pressure from each touching finger is forwarded to the next application layer to be used as control information.

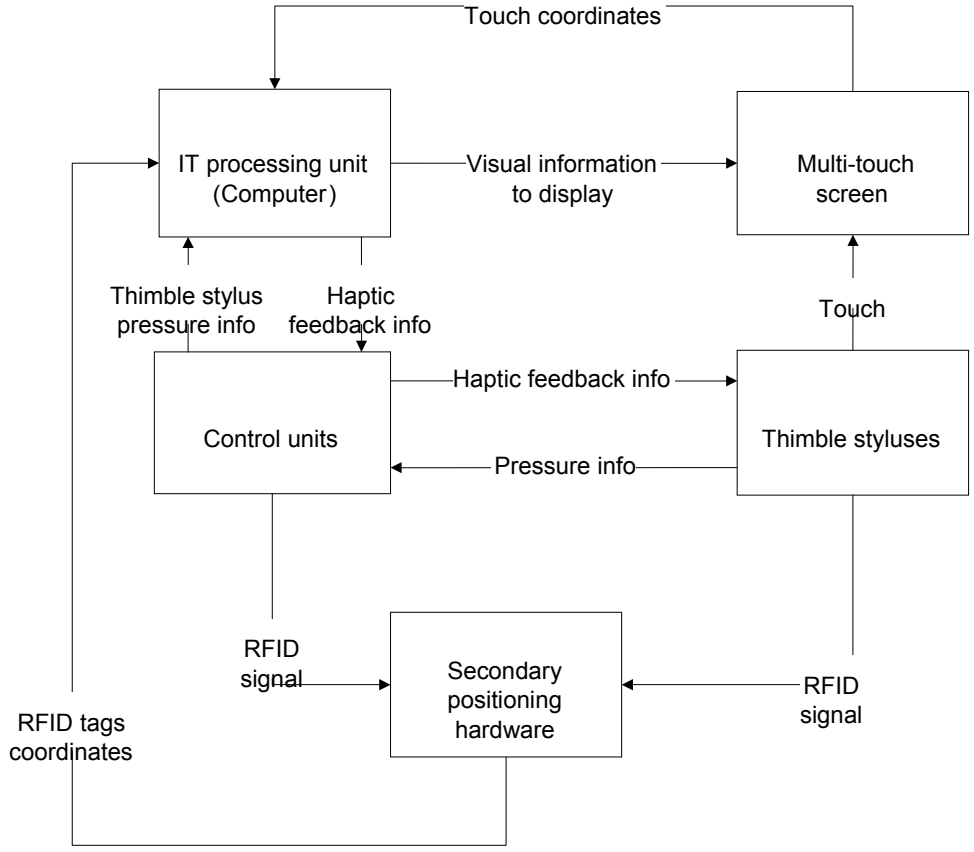


Figure 3.9 The data flow

3.5 Conclusion and future directions

"Everything is best for something
and worst for something else.
The trick is knowing what is what,
for what, when, for whom, where,
and most importantly, why."
[Buxton 2008]

The advantages of the above described solution are: wider communication channel allowing several simultaneous input actions and the use of other senses (for feedback) besides traditional audio/video. The last one is especially important to users with visual disabilities.

Today multi-touch user interface control functions are integrated into all new operating systems, which gives the described here haptic feedback device a good base for implementation and use.

Possible applications of the proposed device are:

- **CAD** – being able to directly manipulate the objects of the design without need to mention several objects simultaneously gives a totally new dimension to the computer aided design process.
- **Data management** – as the amounts of data became larger every day, there is need for newer tools to manage them. Using all ten fingers on a larger surface helps increase productivity by allowing more degrees of freedom to manipulate different data objects.
- **Copy-paste or other functions used in user interface** – when haptic feedback is introduced to the operating system, the copy-paste function can be enhanced so that it better simulates moving objects in real world: if user copies an object with a finger, the textural sense stays in the haptic feedback matrix after the finger has been lifted from the screen, therefore giving the illusion of the object being hold ("clued to the finger") until it will be pasted back to the digital environment.
- **Small devices and wearable computing** – with small devices like palm computers, mobile phones and –terminals there is a common problem with the control panel being too small. With before described finger mounted pointing devices it will be possible to use almost any surface as an input panel, assuming that the touch positioning mechanism will be appropriately re-designed.

- **Virtual control panels** – as the texture on the surface becomes virtual, it will be possible to use a empty tablet and generate new and different input controls like touchable buttons, sliders, etc. for every application, just the way the application designer meant them to be.
- **User interfaces for blind people** – using the before described system it will be possible to design a user interface that allows visually disabled people to sense what is displayed on the screen and manipulate with the objects or even read the text on the screen with the help of Braille code.
- **Gaming and entertainment industry** – By giving users another communication channel with virtual reality vastly enhances the user experience and makes virtual reality more realistic.

4 Optimizing the positioning process

All FTIR and Rear DI based multi-touch and most gesture recognition systems rely on video feed captured by a digital video camera. That stream is then processed using object recognition algorithms to position and track hands, fingers or some tags attached to the fingers. These tags produce light and appear as bright blobs in the video; tracking these blobs allows positioning of tags and together with them – fingers and hands. The position info is then processed into a form of control information for the application program.

In this chapter we describe a method that was developed in order to increase the efficiency of blob tracking process. The approach is similar to the one we used for Tactile Feedback Device in chapter 3. With tactile feedback device we limited the feedback area to the size of fingertips, here we define limits inside each video frame that gets processed by blob tracking algorithm.

This method can also be used in FTIR and Rear DI based systems.

4.1 Introduction

With applications using large displays (e.g. gigapixel screens [König 2007] or large, high resolution FTIR based multi-touch tables [Ideum 2009]) and high resolution cameras appears problem - the amount of information to be processed rises rapidly when camera resolution increases (Table 4.1)

The figures in Table 4.1 were been calculated assuming that each pixel of a colour camera is represented by 3 bytes (RGB).

Nintendo Wii uses separate hardware controllers to overcome this problem [Lee 2008-1], other systems use dedicated CPU units [Shizuki 2006], [Kaltenbrunner 2008]. Most of the current systems use only an infrared (monochrome) camera. There the amount of information to process is only one third of the dataflow presented in Table 4.1.

Using colour camera significantly enhances the possible functionality of the applications, allowing tagging fingers with different colour light emitting diodes (LED's). This allows to identify individual fingers (every finger has different colour), hands (left-right have different colours) or even users (every user has his/her own individual colour) in multi-user environments. Because LEDs are highly monochromatic, emitting a pure colour in narrow frequency range (20-355nm [Knightbright]) many of them can be used simultaneously and still detected individually.

Here is presented a method to improve the blob detection program performance and reduce CPU overhead when running it to detect and track colour (or infrared) LEDs from real-time video feed. Blob here means a closed area in video frame that consist of adjacent pixels and meets predefined criteria's concerning: brightness, colour, minimum and maximum size. The blobs in video frame represent images of

LED's (very bright and sharp spots) or touch points in systems based on frustrated total internal reflection (FTIR). In the field of computer vision, an expression „blob detection" refers to a process that's purpose is to detect and locate points or regions in the image that are either brighter or darker than the surrounding. Blob tracking means repeating the blob detection process over fixed intervals of time and thus determine the locations of the blobs over a certain period of time.

Table 4.1 Dataflow in correlation with camera resolution

Resolution	**Frame rate**	**Dataflow Mbyte/s**
352x288 (standard webcam)	25	7,25
640x480 (good webcam)	25	21,97
800x600 (SVGA)	25	34,33
1024x768 (XVGA)	25	56,25
1280x800 (WXGA)	25	73,24
1280x1024 (SXGA, typical resolution for 17" 4:3 aspect ratio LCD screen)	25	93,75
1680x1050 (WSXGA+, typical resolution for 19-22" wide LCD screen)	25	126,17
1920x1080 (1080p HD, also known as "Full HD")	25	148,32
1920x1200 (WUXGA, 24" Wide LCD screen standard resolution)	25	164,8

4.2 Method

Since blob detection is a part of user interface layer, this process should take as little CPU recourses as possible, in order to leave the majority of calculation power to the main applications.

Presented here method, which is again based on idea of minimizing the information flow only to essential information, allows reducing (compared to the algorithm, described in 4.3.1, without optimization method applied) the amount of time and CPU recourses spent for blob detection calculations up to 40-50%. It was initially developed and tested for an input system that employs one glove with 3 colour LEDs (red, green and blue) and a camera. System diagram is presented on Fig. 4.3

In 4.3.1 is described a “straight force” method for blob detection and 4.3.2 – a method to improve the process. The method, presented in 4.3.2 allows reducing (compared to the algorithm, described in 4.3.1, without optimization method applied) the amount of time and CPU recourses spent for blob detection calculations up to 40-50%. It was initially developed and tested for an input system that employs one glove with 3 colour LEDs (red, green and blue) and a camera. System diagram is presented on Fig. 4.1

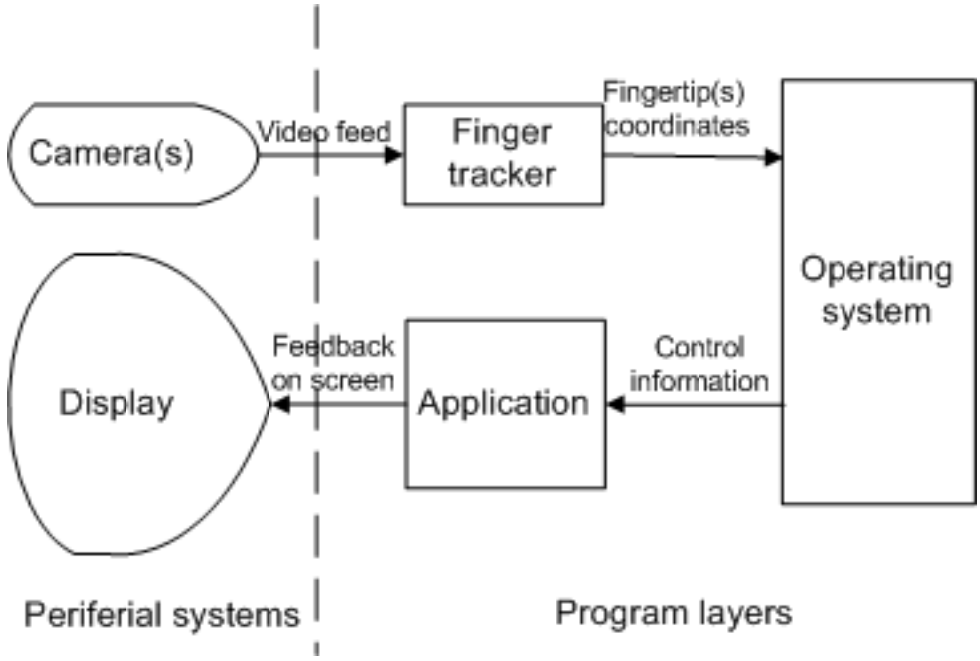


Figure 4.1 Diagram of the finger tracking system

The improvement presented in 4.3.2 is an envelope type of method – i.e. it is wrapped around the actual algorithm, controlling its input and output data, but it does not actually change any code to the blob detection algorithm. In actual testing some modifications were needed for the blob detection algorithm code, but those

code fragments dealt with performance analysis and did not change any logic for the algorithm. Therefore when we address “unmodified algorithm” – this means algorithm code (described in 4.3.1) with performance measurement, but without optimization wrapper.

Testing with real-time video stream from low resolution(352x288 @ 25fps, RGB) webcam revealed that executing blob tracking algorithm (algorithm from chapter 4.3.1) without optimization envelope created a single core utilization of 23%. With better camera (and higher dataflow, Table 4.1) the processor overhead would have been even bigger. This indicates a clear point, where improvement and optimization can be achieved.

4.3 Blob detection and limited track frame method

4.3.1 Blob detection and extraction algorithms

The blob tracking works as follows (Figure 4.2).

When a new frame is received it:

1. Scans the frame row-by-row to find pixels that could be part of the blob(s). The results of this scan, blob candidates, are recorded in array called: *Code Matrix*.
2. Next scan will be based on the *Code Matrix* and it will determine which blob candidates can be merged into blobs. Those blobs are recorded into an array called: *Blob Vector*.
3. Finally the *Blob Vector* gets processed to: eliminate the blobs that don’t meet all the criteria set by blob detection parameter list. And for the remaining blobs size, centre point is calculated, the colour is identified and recorded.
4. Then the process exits with a pointer to the *Blob Vector* that holds an array of remaining blobs with the essential parameters (size, centre and colour).

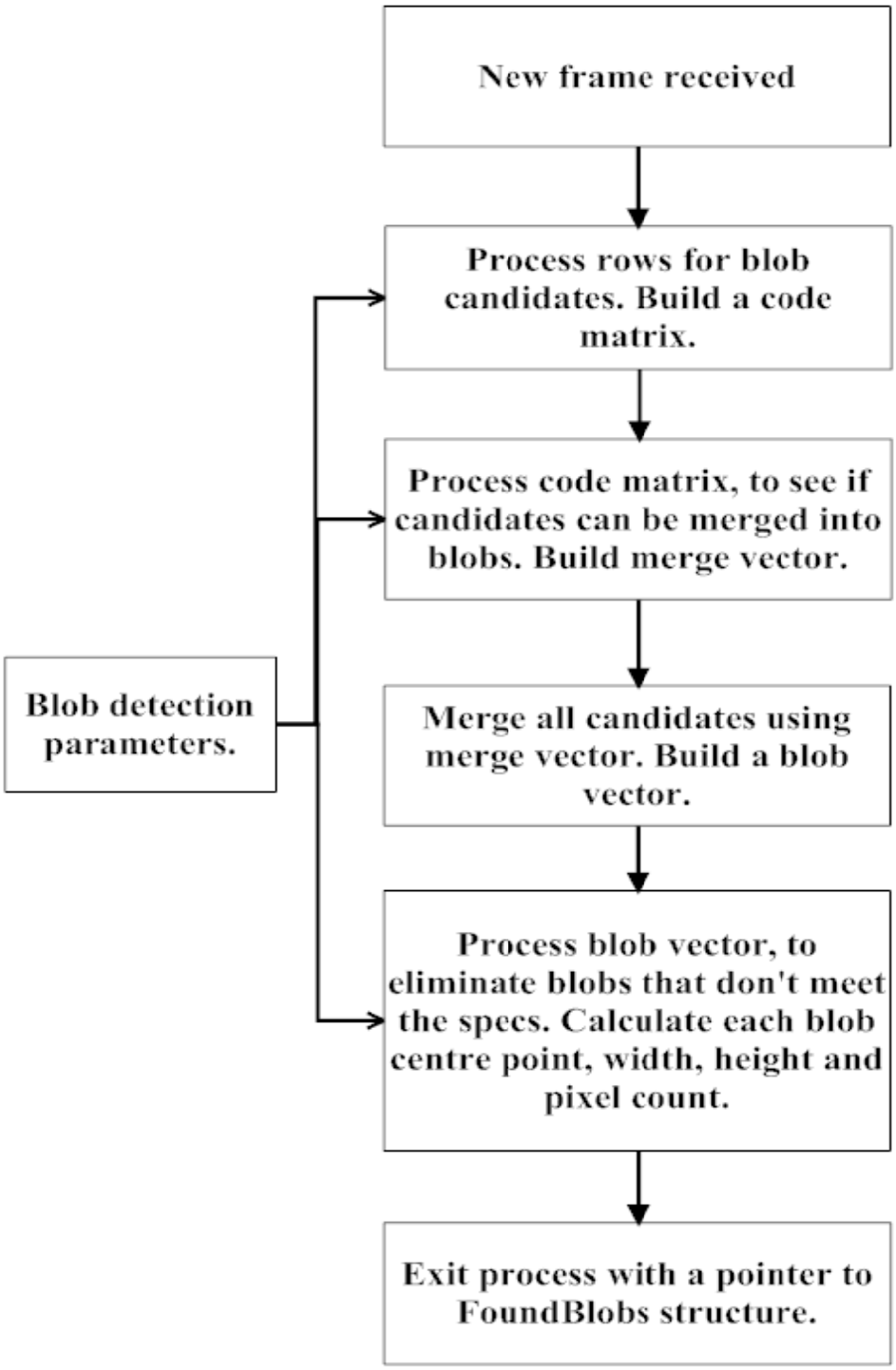


Figure 4.2 Blob detection algorithm

4.3.2 Improving the process – The Limited Trackframe method

The objective was to improve efficiency of the blob detection and tracking process. This is very essential in case of large resolution cameras that output high bit rate video feed to process.

Since the method is not dependent on the algorithm and the algorithm itself has to be supplemented with performance analysis parts and also since in the current application, the LEDs that were tracked left very bright spots which were fairly constant in shape and size in the video stream what makes the detection process simpler, it was decided not to use any existing pre-coded algorithms like CVBLOBLIB [OpenCV], but recreate the code from scratch based on two-pass algorithm by Shapiro [Shapiro 2002] and raster scan by Dave Grossmann [Grossmann].

The finger tracking was implemented by using a glove with colour LEDs on fingertips (Figure 4.3) and then applying blob detection algorithm (described in chapter 4.3.1) to the frames captured by a webcam.

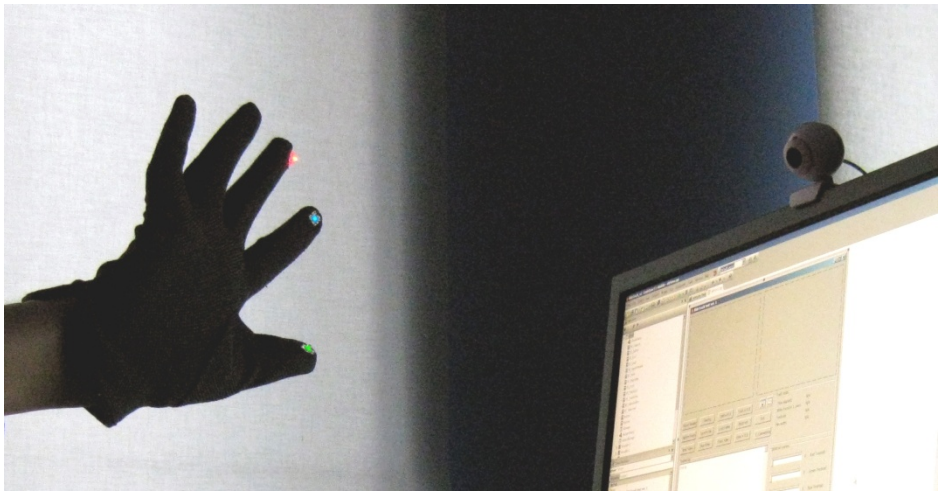


Figure 4.3 Glove with LED's and the camera

In order to assess the algorithm and optimization method several counters were added into algorithm code. They counted when a pixel was accessed in video image and also how many read and write operations were done with the code matrix. First tests were done on still images to train the search parameters. The test indicated that it takes about 4 times more loops than there were pixels in the video frame and ¾ of those were loops accessing or processing the code matrix.

Test results lead to idea, that if the search area could be limited, there will also be less processing. The fact that all LEDs that were tracked, were attached to one hand fingers means that the LEDs can't physically go too far from each other, i.e. the essential information is limited to an area approximately covering the hand's position. Therefore a limited search frame (we call it "trackframe") could be

applied. When the hand and fingers move, that frame should dynamically change its size and location. When some targets get lost (due to tracking error, out of camera vision range or for any other possible reason) the frame should be able to reset itself in order to allow full screen blob detection again.

Based on those ideas, the following wrapper method was created (Fig. 4.5):

- After receiving a new frame from camera, the program checks if trackframe parameters are already defined. If not, then the blob detection will be run on full frame resolution.
- After the blobs have been detected, the first check is if all the blobs were found. In current project this meant that all 3 LEDs (red, green and blue) were found. If not, the trackframe is set to the same size as the video frame.
- If all the blobs were detected, the program calculates the size and location of a rectangle (trackframe) that would fit all 3 blobs.
- Then the trackframe will be increased by predefined percentage to fit in possible hand/finger movements before the next frame is captured. In the test version, this predefined percentage is manually entered, later, if we find a good use for this method, this variable should be dynamically adjusted according to the feedback from tracking result analysis.
- Now a check-up will be made to determine if the increased trackframe still resides inside the video frame and if it exceeds, its size will be corrected accordingly.
- The trackframe painted around detected blobs is presented on Fig. 4.4.

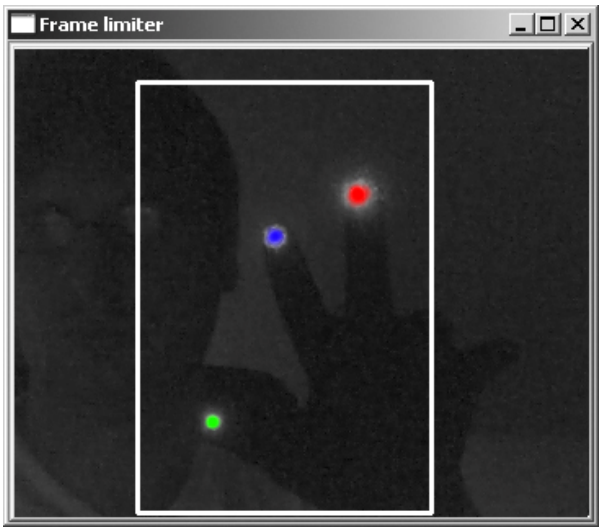


Figure 4.4 Trackframe painted around LED's on captured image

This is a rather basic implementation of the limited trackframe method, but we considered it to be enough to test its effect on the performance of the blob detection algorithm described above.

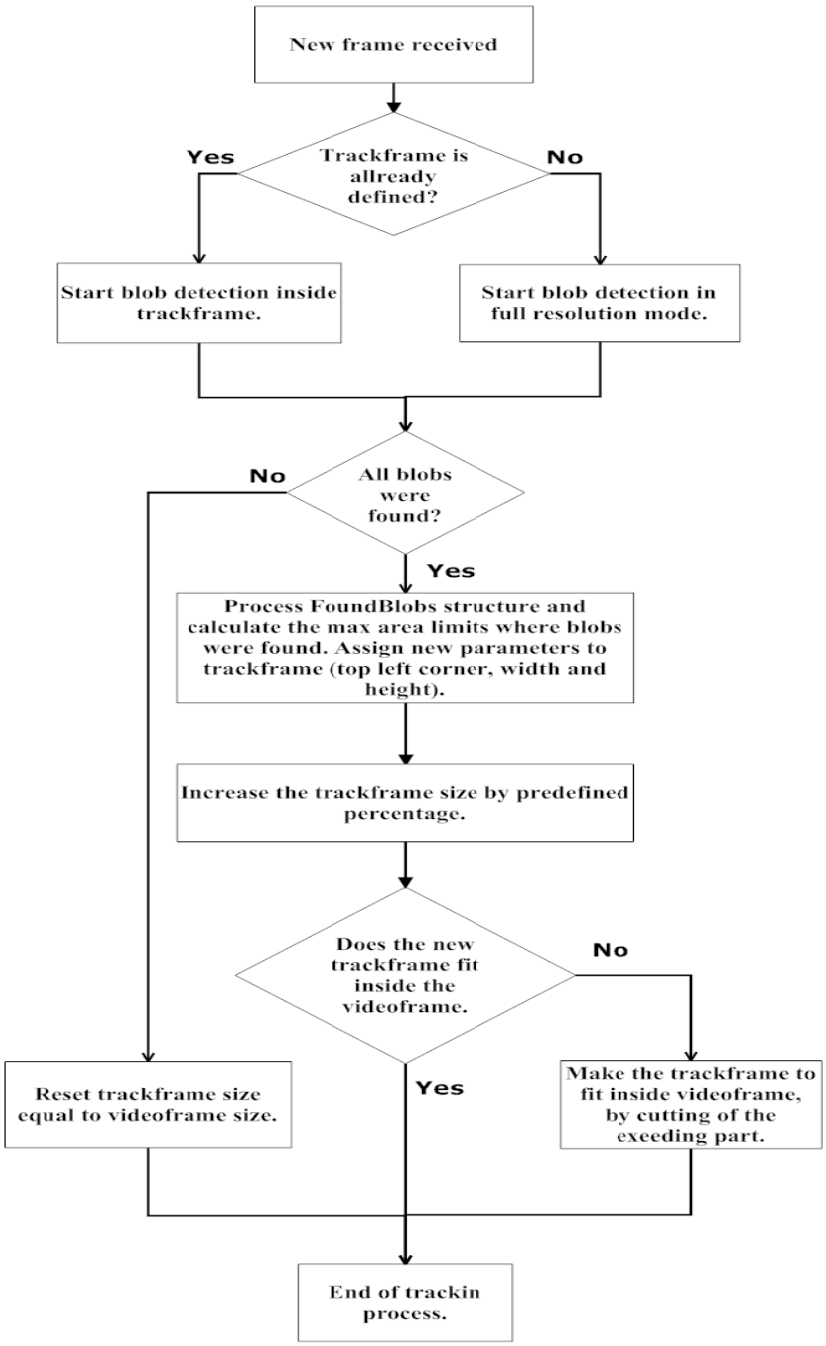


Figure 4.5 Limited Ttrackframe method

4.4 Testing

The success of testing is based on the methods used. Usually performance of a reactive software application is tested on two main dimensions: responsiveness and scalability [Gan 2006].

- Responsiveness is the ability of a reactive system to meet its objectives for response time or throughput. The time interval between two consecutive frames in 25fps video stream is 40ms. In our case the tracking should be finished far before the next frame in the video stream arrives, in order to leave time for the system to process other tasks e.g. run the main application program**.**
- Scalability is system's ability to sustain its response time if demand for software functionality increases. In the current project this is program's ability to cope with video streams with larger resolutions.

4.4.1 Environment

In order to implement the algorithm, optimization method and the testing environment, the next hardware and software components were used:

- A PC with 2,4 GHz Core2Quad processor and 2GB RAM running Windows XP operating system.
- Logitech QuickCam Express USB web camera for 352x288/25fps video stream.
- A4tech Warrior LiveCam for 640x480 still image test.
- Custom made data glove with 3 LEDSs and battery unit attached (Fig. 4.6).
- Codegear C++ Builder.
- OpenCV computer vision function library [OpenCV].

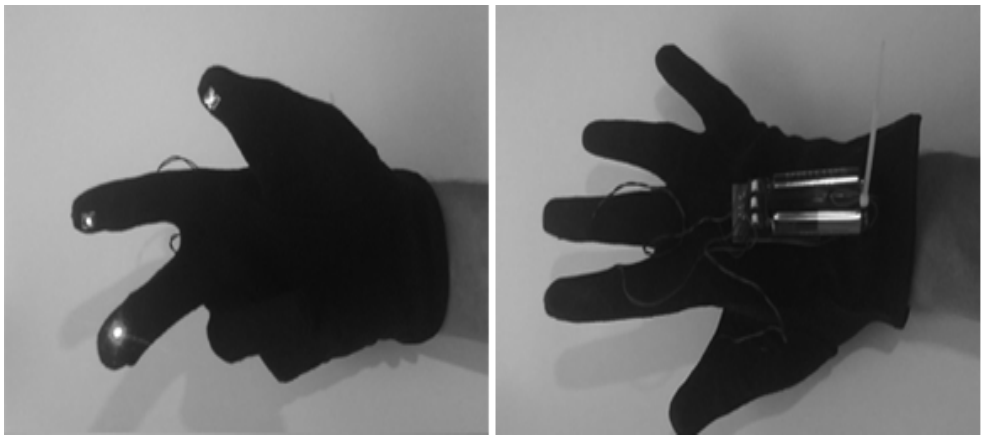


Figure 4.6 Testing glove with colour LEDs

4.4.2 Test implementation

For the current project were used three test strategies:

- Measuring the time that is spent to process one frame.
- Measuring the number of cycles the algorithm performs in average during one second of live video stream.
- Measuring the CPU and memory usage during program execution with Windows task manager.

The first two strategies required some modifications in the program source code. Since first strategy required timing, a timer function was implemented in the test environment. To get more accurate results and average the effect of other running processes on the PC the blob detection algorithm was executed 25 times a row using a still image and the total time was divided accordingly.

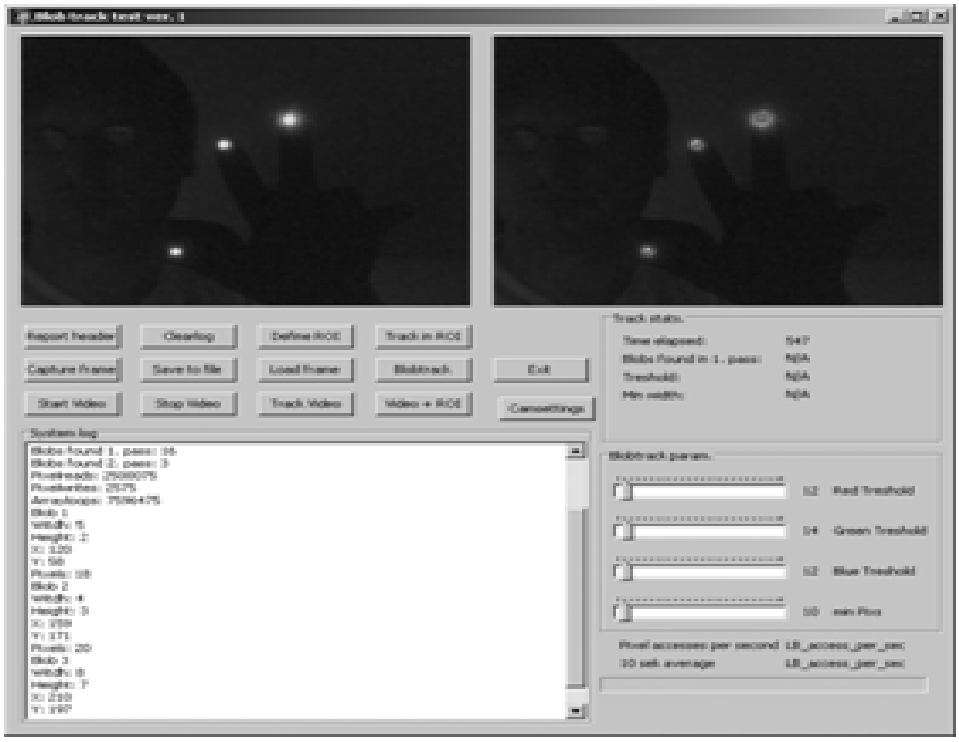


Figure 4.7 Test program for the Blob detection process optimization

For the second strategy counters were placed inside certain functions. Two different counters were used:

- Pixel read/write in image array.
- Code matrix element access.

The values of those counters were recorded in log each second and then reset. From those log files average values were then calculated.

The screen of the testing and performance evaluation program is presented on Figure. 4.7

4.4.3 Test plan

The main purpose of the test was to prove the effect and stability of the program code that was created using the above described algorithm. Based on test strategies and implemented measuring methods the next tests were planned (Table 4.2)

Table 4.2 The test plan

Test no.	Test subject	Resolution	Measuring unit
1	Still image	352x288	Time
2	Still image	640x480	Time
3	Video stream	352x288 @ 25fps	Cycles
4	Video stream	640x480 @ 25fps	CPU usage

All 8 tests were performed in similar conditions with and without using Limited Trackframe method.

4.4.4 Results and analysis

In this chapter are presented the test results and comments.

In table 4.3 are results for still image tests. If the video stream frame rate is 25fps, then the time between two consecutive frames is 40 milliseconds. This means that the current algorithm can be run real-time on 352x288 @ 25fps video stream, but not on 640x480 @ 25fps. Therefore the next tests involving real-time video stream were executed on 352x288 @ 25fps only.

Table 4.3 Still image test results

Test no.	Resolution	Processing time (ms)	
		Full Frame detect	Limited frame detect
5	352x288	23,12	11,24
6	640x480	72,48	48,12

In table 4.4 are presented values of cycle counters (they were explained under "Test implementation"). Full frame detection performed 7.7 million cycles per second i.e. 308 thousand cycles per frame. Considering that every frame on 352x288 resolution has 101376 pixels, this translates to approximately 3 cycles per every pixel. After the dynamic trackframe was activated it came down to approximately 1.29 cycles per pixel, meaning that the same tracking result was achieved more than two times quicker.

Table 4.4 Video stream test results

Test no	Resolution and frame rate	Thousand processing cycles (per second)	
		Full frame detect	Inside frame detect
7	352x288 @ 25fps	7708	3226

In Table 4.5 are presented average CPU usage test results. The program was written without special optimization methods or multithreading technologies. This allowed to simplify the performance measuring. Therefore the numbers in Table 4.5 mean single core usage (despite the fact that the PC was running a quad core processor.)

Table 4.5 Video stream test II results

Test no	Resolution and frame rate	CPU usage	
		Full frame detect	Inside frame detect
4	352x288 @ 25fps	23%	15%

The performance gain displayed in table 5 is slightly less than 2 times. This can be explained with the fact, that the program was performing other tasks as well, which were not affected by the optimization method. Those tasks included video capturing from the webcam through USB port, presenting source feed and delivering results on display screen in real-time.

4.5 Conclusions and future directions

Blob detection and tracking plays important role in several multi-touch and gesture recognition input systems. Therefore it is essential that it executes as fast as possible using as few CPU resources as possible. Described here method allows, according to the test results, reduce CPU usage ~50%.

But every big advancement has some setbacks. Next are presented the positive and negative features of the method:

Positive:

- It is simple to apply.
- It gives good results (ca 50% improvement on test environment, with 352x288 px camera, but with greater resolution cameras the gain will increase, as the physical tracking area is bigger, but the tracframe to follow remains the same - size of the hand).
- It is easily scalable for video stream with larger resolution.

Negative:

- At the moment it is designed for a specific application (constant number of blobs, i.e. 3 LEDs to track) and the effect will likely decrease if the number of blobs to detect is increased.

The next steps in development of this method would be to make it work with variable number of blobs. Also controlling the size of the trackframe has lot of room for improvement, e.g. introduce dependence between trackframe size adjustment percentage and trackframe reset frequency (described in p4.3.2). Another improvement could be using multiple trackframes (one for every blob) with full scale detection after certain number of frames. The final objective is to make blob tracking in higher resolution video streams more efficient so it could be run real-time, without overloading the CPU.

There is also possibility to integrate the method with Tactile Feedback Device's secondary positioning hardware (described in p3.3.5), so that the radio location based tracking defines the locations for limited search windows.

Currently such an integration can be realized only in virtually modelled environment, thus the real benefits can be assessed only when a working prototype of the tactile Feedback Device has been manufactured.

5 System usage

Invention is worth nothing,
if nobody is able to use it ...

5.1 Introduction

In this section is described the different usage fields of haptic feedback user interface system. Sub-section 5.4 is largely based on a publication "Employing Haptic Input-Output for Cognitive retraining Applications", that was co-written with A. Mellik and J. Tulviste and presented on VECIMS conference in 05.2009 [Joasoon 2009].

Although the technology can be used and implemented in various fields, due to the partnership with Cognuse OÜ and ongoing research we concentrate mainly on the medical field, more specifically: cognitive retraining and rehabilitation techniques.

5.2 Enhanced interface

As mentioned earlier the purpose of tactile interface is to communicate information through the sense of touch, allowing people to feel the virtual objects. This is an area of growing interest in the research community [Hafez 2007].

The application groups benefiting most from tactile interface technology are as follows [Benali-Khoudja 2004]:

- Teleoperation and telepresence;
- Sensory substitution;
- 3D surface generation;
- Braille systems;
- Games.

Telepresence is a technology, which allows a person to feel as they were or give the appearance as if they were present at the location other than their true location. Telepresence requires that senses of the user are provided with stimuli that originate from the other location. As a minimum it usually involves video and audio as the easiest types of information to capture and transmit. In order to give the user ability to manipulate with objects in remote environment a tactile feedback is essential. Working in hazardous environments, remote surgery (Figure 5.1), educational telepresence and art are just a few subject fields to mention [Riva 2003].

Sensory substitution means transforming the characteristics of one sensory modality into stimuli of another sensory modality. It was first introduced in 1960's by Paul Back-y-Rita in order to use one sensory modality, mainly tactition, to gain environmental information to be used by another sensory modality, mainly vision.

When a person becomes blind they don't generally lose the ability to see, they only lose the ability to transmit the ability to receive the sensory signals from the periphery (retina) to the brain. Since they still have vision processing pathways , they can still "see" the subjective images by using other sensory modalities like touch or audition [O'Regan 2001].

Figure 5.1 Telepresence system for training session of remote surgery (picture from express.howstuffworks.com, courtesy of NASA)

In 3D surface generation we mean applications that generate 3D information for objects on the display screen. That information is then used to generate a 3D tactile surface. The surface can be physical, in which case a tactile display screen is needed and the surface is felt with bare fingers or virtual in which case the tactile sensation is created directly on the fingertips with help of special hardware like the Tactile Feedback Device, described in this thesis.

Purpose of a Braille system is to enable visually disabled people to read and write. Devised in 1821 by Louis Braille the system works solely on the sense of touch. Each Braille character or cell is made up of six dots, arranged into two columns of three dots (Figure 5.2). The Tactile Feedback Device can successfully emulate the Braille displays (described in chapter 2.4.2).

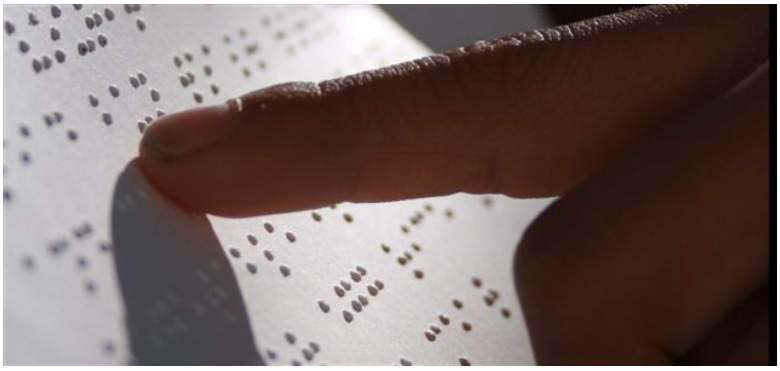


Figure 5.2 Braille text (picture from Braillewithoutborder.coms)

The gaming industry is probably the most active research group in developing all kind of alternative ways to make the gaming experience better. Force feedback joysticks and steering wheels have been around for some time. Tactile Gaming Vest was displayed on IEEE haptic symposium 2010 [IEEE VR 2010].

Figure 5.3 Tactile Gaming Vest (picture from singularityhub.com, courtesy of Saurabh Palan)

Transferring virtual game objects into touchable and sensible ones would be a major step for game industry.

5.3 Interface for visually disabled

Due to the established partnership with Cognuse OÜ and planned longer term research activities, the visually disabled users were targeted as our main audience for the technology development.

The scope of the research is to develop an enabling technology and not specific applications. Therefore we can here give only guidelines, how the technology could be implemented in order to solve problems with current technologies and interfaces for visually disabled people.

The interfaces for visually disabled can be divided into two main classes (Figure 5.4):

- Text bases interfaces
- Graphical object based.

In text based interfaces the communication is driven by linguistic information which is communicated via auditory senses or touch (using Braille code.)

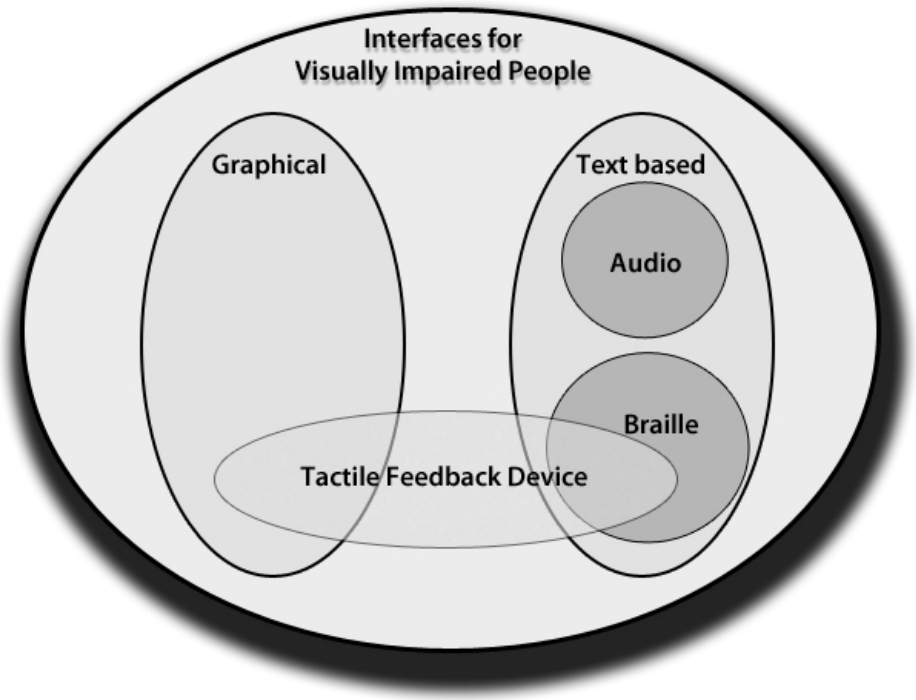


Figure 5.4 Segmentation of Interfaces for Visually Impaired People

The graphical interface uses sensory substitution (described in chapter 5.2).

The discussed in this thesis Tactile Feedback Device can work in graphics mode, in which case the objects will be converted into a tactile surface under the users' fingertips but it can also emulate the Braille code.

5.4 Usage in cognitive retraining process and rehabilitation

5.4.1 Background

This subsection describes the plans on using the Tactile Feedback Device in the field of cognitive learning and rehabilitation. It is based on ongoing preliminary research done in partnership with Tallinn Technical University Mechatronics Department, Cognuse OÜ and Jukulab OÜ.

Cognitive retraining is a therapeutic strategy that seeks to improve or restore a person's skills in the areas of paying attention, remembering, organizing, reasoning and understanding, problem-solving, decision making, and higher level cognitive abilities [Encyclopaedia of Mental Disorders].

Cognitive retraining is used for people with cognitive problems associated with brain injuries, different disabilities or disorders and aging. The main purpose is to reduce the everyday problems for people with mentioned difficulties and therefore improve their life quality.

Cognuse OÜ and Jukulab OÜ have developed a cognitive retraining system called BrainCapsule (Figure 5.5). Brain Capsule is a hardware platform for running computer based cognitive retraining and diagnostics programs. It is meant to use by general purpose health institutions, SPA's and wellness centres [BrainCapsule].

Figure 5.5 BrainCapsule concept (picture from www.cognuse.com)

Cognitive skills in human beings are generally divided into following groups (Figure 5.6) based on the underlying function or cognitive function or domain e.g.:

- Attention;
- Memory;
- Language;
- Decision-making skills, planning etc;
- Visio-spatial skills.

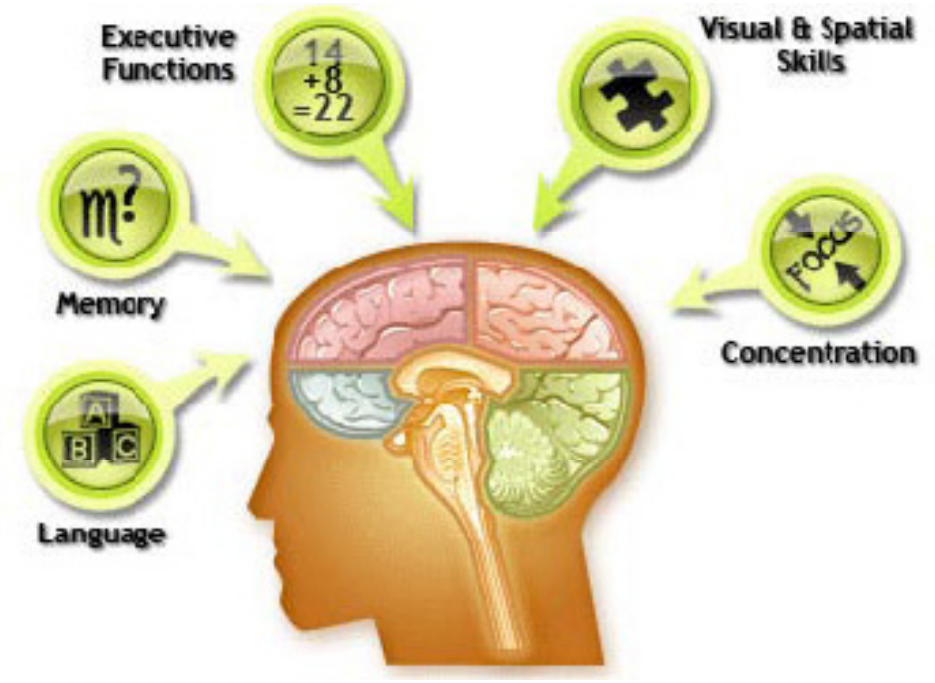


Figure 5.6 Cognitive skills of a human brain (picture from www.cognuse.com)

5.4.2 Using Virtual Reality (VR) and Haptics in rehabilitation

The trend to use VR in rehabilitation process gives the potential to create systematic human testing, training and treatment environment that allows precise control of complex dynamic 3D stimulus presentations. Inside that environment sophisticated interaction, behavioural tracking and performance recording with analysis is possible. However, this trend is commonly limited to audio-visual VR systems that do not provide haptic feedback [Feintuch 2006].

Clinical-level computer-based cognitive retraining applications have traditionally employed a standard personal computer with its standard user interaction devices (display, loudspeakers, keyboard, and mouse). Since the process is at most times overseen by a professional neuro-psychologist, who handles appropriate help requests and directs the user through the often rigorous training regime, any improvements regarding user interface design or screen layout are likely to carry a noticeable effect to the rehabilitation process outcome and desired end result.

Introduction of haptic technologies to cognitive rehabilitation allows introduction of novel dynamic feedback systems regarding perception and touch modality physical sensations. Haptic sensory feedback compliments the Braille system by allowing real-time, dynamic dot combination presentations. Thus, dot arrangement sequences can be perceived without finger movement. Furthermore, dynamic input

features are an extension to the use of compensatory mechanisms in case of blindness or damage to the visual cortex pathways, allowing for novel feedback features via touch and perception pathways [McLaughlin 2005].

Haptic technologies potentially contribute to rehabilitation efforts in any of the aforementioned domains by addressing the domain in a direct manner, via a compensatory route or as a part of a holistic rehabilitation programme. Attempts to integrate haptic systems into various neuropsychological treatments and post-stroke rehabilitation have indicated promising results [Alamri 2008].

5.4.3 Using Tactile Feedback Device in Cognuse OÜ applications

Visual images on a computer display and auditory aids in form of clear audio signals are used in most of cognitive retraining applications (Figure 5.7), ultimately limiting the target user group. Visual or auditory aids fail to supplement the training applications for patients with damage to cortical vision or auditory pathways as well as for patients suffering from damage directly related to the eye or ear.

If need for clinical cognitive rehabilitation is established, it is important to start the training process as soon as possible. Due to visual or auditory impairment caused by the traumatic events to the person, the delivery of rehabilitation program exercises might be delayed (until the auditory or visual impairment is recovered) or terminated, thus leading to a failure to fully exploit opportunities for recovery via post-traumatic direct or compensatory skill training.

Furthermore, patients with pre-existing impairments are automatically excluded from the target group, as in its standard form the intended functionality of the application is rendered undeliverable to the patient (due to the auditory or visual impairments).

Our work intends to broaden the target audience for the already existing and new cognitive retraining (and other) applications and automatically adapt the existing training tasks to focus on alternative cognitive or motor skills and senses. Therefore, if a person has serious and possibly irreversible impairments regarding his or her auditory system (cortical or ear-based), the feedback will be delivered in a miniscule force-feedback manner.

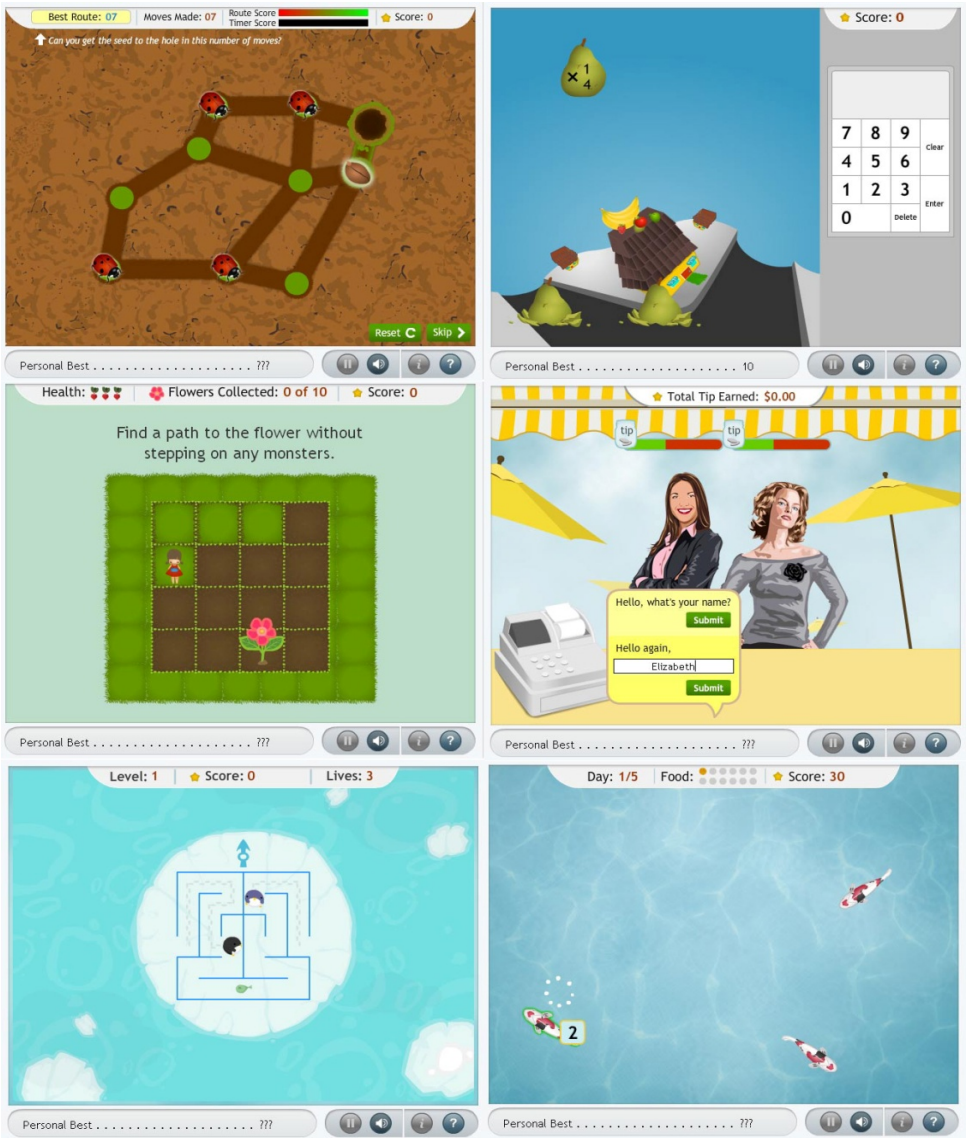


Figure 5.7 Cognitive retraining applications (picture from www.cognuse.com)

Table 5.1 introduces the general idea behind mapping functions within the cognitive domain. In its current form it only describes the opportunities presented by our method and is pending review by clinical neuro-psychologists.

Table 5.1 Function mapping

Impairment	**Functions**		
	Original function	**Transferable?**	**Proposition**
Hearing	Audio signal generated	YES	Haptic feedback
Visual	Small image displayed	YES	Iconized or segmented for haptic feedback
Visual	Large image displayed	NO	
Visual	Text displayed	YES	Braille code to haptic output

Cognuse OÜ intends to explore the opportunities further in order to tie the current concepts to seamlessly interact also with web-based applications. This requires an integrated framework, where standard web-applications can access the local hardware resources (as Flash-components are able to access one's web-cam, microphone, etc). Several suitable software environments are becoming available (like AdHapticA [Orozco 2008]) so that an appropriate one can be selected.

6 Conclusions

6.1 Overview

Different virtual reality technologies allow users to interact with synthetic worlds that are purely made up from data. In this thesis we described an enabling technology that allows users to touch and feel those virtual objects.

The project started up with the need to improve the communication channel between man and computer. In order to make this improvement we studied the main direction of HCI, researched the existing technologies – what are their strong sides and weak ones. Based on this information a new apparatus, the Tactile Feedback Device was designed together with the methodology how to make it operate and how to link it with pre-existing systems. The design was guided by communication minimization and localization principle – the device should present only local information which is directly perceivable to human user.

In order to verify that our invention had unique properties, a patent claim was filed on 04.01.2007. Estonian Patent Agency confirmed the uniqueness and issued a patent certificate on 14.12.2008. The technology was published at IADIS conference on 2007 [Joasoon 2007].

At the time, when the tactile feedback device was handled by the Patent Office, we considered another problem – how to reduce the CPU utilisation in finger positioning when processing the video feed from positioning cameras; this problem concerns all FTIR and Rear DI systems. The information flow minimization principle turned out to be very useful also here.

The developed Limited Trackframe method proved to be very promising. The testing showed positive effect on reducing CPU overhead up to 50%. This research was also published in IEEE conference [Joasoon 2008-2]

Considering that at the moment FTIR and Rear DI methods are most cost effective ways to create large scalable multi-touch environments, the Limited Trackframe method has clearly proved its importance and practical applicability.

In 2008 we found some research partners interested to use the Tactile Feedback Device in one of their projects involving Cognitive retraining. The early concept of two technologies working together was published in VECIMS conference 2009 [Joasoon 2009] and that was followed with a preparation of pre-research project later in 2009. In 2010 we were able to start an official EAS funded pre-research project, to find answers to the following questions:

1. What are the current tactile feedback providing technologies and how do they compare to our Tactile Feedback Device for Multi-touch User Interfaces.
2. What are the necessary base technologies in order to build a working prototype.

3. What are the possible user groups, who can benefit from this device and what competitive technologies they currently use.
4. What is the current market situation for haptic technologies and what should be the business model and marketing strategy to sell this technology.

6.2 Contributions

Contributions of this dissertation can be summarized as follows:

1. A new method to provide detail tactile feedback in multi-touch user interfaces.
2. A Tactile Feedback Device to implement the new method. The uniqueness of the method and the device was verified with the patent EE 05116 B1.
3. A method to improve CPU usage in FTIR and Rear DI positioning systems based on high resolution video cameras, where real-time processing of live video feed is required.
4. In cooperation with Cognuse the possibility to provide better cognitive retraining applications for patients with visual disabilities.

6.3 Future research

At the moment we started in partnership of Cognuse OÜ, Jukulab OÜ and Tallinn Technical University's Department of Mechatronics a pre-research to determine the commercial value of the Tactile Feedback Device. The pre-research is funded by EAS and if it gives positive outcome, we can continue with full technology research that concludes with building a first fully functional prototype as a demonstration device to market the technology and also as a research device that allows performing tests on real subjects in order to improve the technology. One possible improvement can be implementation of vibro-tactile feedback in order to overcome the effect of fingertip skin receptors getting adapted to constant pressure and therefore not forwarding the correct sense impulses to the brain [Perez 2000] [Regenbrecht 2005].

There are also possibilities to develop further and modify the Limited Trackframe method :

1. Make it work with variable amount of traceable objects.
2. Integrate it into Tactile Feedback Device two-level positioning system so that the triangulation based tracking process also defines the limited size search windows.

The second possibility can be better researched and developed when we have completed the technology research and manufactured a prototype. Then there is a real equipment to test the necessary algorithms and measure the effect.

We see possibilities to use the Tactile Feedback Providing Matrix in different 3D gesture based interfaces, in order to give the users some feedback/guidance when they make gestures in open air in front of the camera/sensor array.

And finally we also see possibilities to use the developed Tactile Feedback technology in stand-alone wearable computing and guiding devices.

When coupled with proximity sensors, the haptic device may be used to inform the user about possible obstructions at all sides or in case of high resolution image processing from a compact camera image, the surroundings and extracted textual information may be fed to the user's fingertips via Braille code. Furthermore, if applicable, such images covering terrains could be forwarded to the matrix of solenoids(described in p3.3.3 and Figure 3.7).

References

AFB – American Foundation of Blind „What is Braille?“ Retrieved on 28.12.2009 from: http://www.afb.org/Section.asp?SectionID=6&TopicID=199

AIS – National Institute of Advanced Industrial Science and Technology, “PC World Extended for The Visually Handicapped through Touch” Homepage accessed and information retrieved on 26.04.2010 from: http://www.aist.go.jp/aist_e/latest_research/2004/20040622/20040622.html

Alamri, A., Eid, M., Iglesias, R., El Saddik, A. and Shirmohammadi, S. (2008) “Haptic virtual rehabilitation exercises for post-stroke diagnosis”, *IEEE Transactions on Instrumentation and Measurement, Vol. 57, No. 9,* pp.1876-1884.

Benali-Khoudja, M., Hafez, M., Alexandre, J.-M., Kheddar, A. (2004) “Tactile interfaces: A state-of-the art Survey,” Proc. ISR 2004 -35th Intl. Symp. Robotics, Paris, March 2004.

Blob Extraction Library. Accessed and information retrieved on 2008 from: http://opencvlibrary.sourceforge.net/cvBlobsLib

Bradski, G., Kaehler, A. (2008) „Learning OpenCV: Computer Vision with the OpenCV Library“ 1st edition, O'Reilly Media, Inc. 2008.

BrainCapsule – Product webpage accessed and information retrieved on 29.04.2010 from: http://www.cognuse.com/products/braincapsule

Brewster, S., Chohan, F., & Brown, L. (2007). “Tactile feedback for mobile interactions.“ In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems* , (pp. 159-162). New York, NY, USA: ACM.

Burdea, G. C., Coiffet, P. (2003). „Virtual Reality Technology.“ Wiley-IEEE Press, New York, ISBN 0471360899.

Burke, J. L., Prewett, M. S., Gray, A. A., Yang, L., Stilson, F. R. B., Coovert, M. D., Elliot, L. R., & Redden, E. (2006). “Comparing the effects of visual-auditory and visual-tactile feedback on user performance: a meta-analysis.” In *ICMI '06: Proceedings of the 8th international conference on Multimodal interfaces* , (pp. 108-117). New York, NY, USA: ACM.

Buxton, B. (2008) „Multi-Touch Systems that I Have Known and Loved”. Retrieved on 23.08.2008 from: http://www.billbuxton.com/multitouchOverview.html

Cognuse OÜ, Company homepage accessed and information retrieved on 23.04.2010 from: http://www.cognuse.com/

Diaper, D. and Stanton, N. A. (2004) „The Handbook of Task Analysis for Human-Computer Interaction.“ Lawrence Erlbaum Associates, Inc., Publishers, Mahwah, New Jersey, USA.

Eden, J. (2003) “Designing for Multi-Touch, Multi-User and Gesture-Based Systems” in Dr.Dobb’s portal. April 03. 2009. Page accessed and information retrieved on 29.03.2010 from: http://www.drdobbs.com/architecture-and-design/216402697;jsessionid=ZSEI0QVA1RHRPQE1GHPSKHWATMY32JVN

Elliott, R. S. (1999) “Electromagnetics: History, Theory, and Applications” Wiley-IEEE Press. ISBN: 978-0780353848.

EnableMart UK Ltd. Company homepage accessed and information retrieved on 26.04.2010 from: http://www.techready.co.uk/

Encyclopedia of Mental Disorders. Webpage accessed and information retrieved on 23.05.2010 from: http://www.minddisorders.com/index.html

Fact Monster, Page accessed and information retrieved on 29.12.2009 from: http://www.factmonster.com/dk/science/encyclopedia/skin.html

Feintuch, U., Raz, L., Hwang, J., Josman, N., Katz, N., Kizony, R., Rand, D., Rizzo, A. S., Shahar, M., Yongseok, J. (2006) “Integrating Haptic-Tactile Feedback into a Video-Capture-Based Virtual Environment for Rehabilitation” published in*: Cyberpsychology and Behaviour. Vol 9; Issue 2; pp 129-132.* Mary Ann Liebert, Inc. USA. ISSN: 1094-9313

Favalora, G. E. (2005) Volumetric 3D Displays and Application Infrastructure. *Computer vol.38,Issue 8 (Aug. 2005)*, (pp. 37-44.) IEEE Computer Society Press Los Alamitos, CA, USA. ISSN:0018-9162

Fujieda, I. and Haga, H. (1997) "Fingerprint input based on scattered-light detection", Applied Optics, Vol. 36, Issue 35, pp. 9152-9156, 1997.

Frey, M. (2009) “SnOil - A Physical Display Based on Ferrofluid” published in Klanten, R., Ehmann, S., Huebner, M. “Tangible – High touch visuals” Die Gestalten Verlag. Jan 2009. ISBN: 978-3899552324

Friedlander, N., Schlueter, K. & Mantei, M. M. (1997) “Easy tactile feedback at bargain basement prices”, In *CHI '97 extended abstracts on Human factors in computing systems: looking to the future,* (pp. 307-308). Atlanta, Georgia, USA: ACM

Fundamentals of Haptics (Interdepartmental Research Centre "E.Piaggio"), Page accessed and information retrieved on 30.12.2009 from: http://www.piaggio.ccii.unipi.it/index.php?en/157/fundamentals-of-haptics

Gan, X. (2006) “Software performance testing”. University of Helsinki, Faculty of Science, Department of Computer Science, seminar paper 26.9.2006.

Gettys, E. W., Keller. F. J. and Skove, M. J. (1989) “Classical and Modern Physics.” New York: McGraw-Hill, 1989.

Gonzalez. C. R., Woods, E. R. (2002) “Digital Image Processing“ Second edition, 2002 by Prentice-Hall, Inc. Upper Saddle River, New Jersey 07458

Grossmann, D. “Comments on Blob Analysis Library”. Accessed and retrieved on 29.01.2010 from: http://yi-ihpc.blogspot.com/2008/02/dave-grossman-comments-on-blob-analysis.html

Grossman, T., Wigdor, D., and Balakrishnan, R. (2004) “Multi-finger gestural interaction with 3d volumetric displays.” In UIST '04 *Proceedings of the 17th Annual ACM Symposium on User interface Software and Technology*. (pp. 61-70). New York, NY, USA: ACM

Hafez, M. , (2007) "Tactile interfaces: technologies, applications and challenges", The Visual Computer: International Journal of Computer Graphics Vol. 23, Issue 4 (March 2007). Pp 267-272. Springer-Verlag New York. ISSN:0178-2789.

Han, J. Y. (2005) “Low-Cost Multi-Touch Sensing through Frustrated Total Internal Reflection”. In *UIST '05: Proceedings of the 18th annual ACM symposium on User interface software and technology, (* pp. 115-118), 2005.

Han, J. Y. (2006). “Multi-touch interaction wall.” In *SIGGRAPH '06: ACM SIGGRAPH 2006 Emerging technologies* . New York, NY, USA: ACM Press.

Handy Tech Electronik GmbH, Company homepage accessed and information retrieved on 28.12.2009 from: http://www.handytech.de/en/normal/start/index.htm

Hewett,T., Baecker, R., Card, S., Carey, T., Gasen, J., Mantei, M., Perlman, G., Strong, G. and Verplank, W. (1992) „ACM SIGCHI Curricula for Human-Computer Interaction“ 1992,1996 Page accessed on 06.1.2010 from: http://old.sigchi.org/cdg/

Hoggan, E., Brewster, S. A., & Johnston, J. (2008). “Investigating the effectiveness of tactile feedback for mobile touch screens.” In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems* , (pp. 1573-1582). New York, NY, USA: ACM.

Hotelling, S. et al. (2006). „Multipoint touch screen“. United States Patent Application 20060097991

Ideum multi-touch table (2009). Product description, Retrieved on 22.12.2009 from: http://www.ideum.com/products/multitouch

IEEE VR (2010). IEEE Virtual reality conference, MA, USA, 20-26 March 2010. Conference homepage accessed and conference program retrieved on 16.04.2010 from: http://conferences.computer.org/vr/2010/program/IEEE_VR_2010_FinalProgram.pdf

Ishida, T., Shimojo, M. and Watanabe, T. (2000) „Screen reader using mouse with tactile display for blind", 5th Conference of the Virtual Reality Society of Japan, pp.477-478, Tsukuba, Japan, September 2000

Johnson, J., Roberts, T., Verpland, W., Smith, D., Irby, C. and Beard, M. (1989) „The Xerox Star: A Retrospective." *Computer Graphics*, 22(9):11-26, 1989.

Jukulab OÜ, Company homepage accessed and information retrieved on 23.04.2010 from: http://www.jukulab.ee/

Kaltenbrunner, M. and Bencina, R. (2008) "reacTIVision: A Computer-Vision Framework for Table-Based Tangible Interaction", *Proceedings of the first international conference on "Tangible and Embedded Interaction" (TEI07). Baton Rouge, Louisiana*. Retrieved on 23.08.2008. from: http://reactable.iua.upf.edu/pdfs/reactivision_tei2007.pdf

Kaltenbrunner, M., Bovermann, T., Bencina, R., Gostanza, E. (2005) „TUIO: A Protocol for Table-Top Tangible User Interfaces" *Proceedings of the 6th Int'l Workshop on Gesture in Human-Computer Interaction and Simulation* (2005)

Kenshalo, D.R. (1968). „The Skin Senses". Springfield. Thomas 1968

Kim, S., Kim, C., Yang, G., Yang, T., Han, B., Kang, S., and Kwon, D. (2009). „Small and lightweight tactile display(SaLT) and its application." *In Proceedings of the World Haptics 2009 - Third Joint Eurohaptics Conference and Symposium on Haptic interfaces For Virtual Environment and Teleoperator Systems* (March 18 - 20, 2009).

Knightbright Corporation, LED manufacturer. Homepage accessed and LED datasheets retrieved on 29.01.2010 from: http://www.kingbrightusa.com/

Koskinen, E., Kaaresoja, T., and Laitinen, P. (2008). "Feel-good touch: finding the most pleasant tactile feedback for a mobile touch screen button." In *IMCI '08: Proceedings of the 10th international Conference on Multimodal interfaces* (pp. 297-304.) New York, NY: ACM

Kramer, J.F. (2001) „Force feedback and texture simulating interface device". United States Patent Application Publication No US 2001/0043847 A1.

Kurzweil, R. (2001) „The Law of Accelerating Returns" Published by KurzweiAI.net March 7, 2001 Retrieved on 29.05.2009 from: http://www.kurzweilai.net/articles/art0134.html?printable=1

König, W. A., Bieg, H. J., Schmidt, T., Reiterer, H. (2007) “Position-independent interaction for large high-resolution displays”, Proceedings of IADIS International Conference, Interfaces and Human Computer interaction 2007, pp 117-125.

LaMotte, R. H. and Srinivasan, M. A. 1991. “Surface microgeometry: Tactile perception and neural encoding.” *In Information Processing in the Somatosensory System, O. Franzen and J. Westman, Eds. Macmillan,* pp. 49—58, London.

Lederman, S. J. , Pawluck, D. T. (1992) “Lessons From the Study of Biological Touch for Robotic Haptic Sensing,” in Advanced Tactile Sensing for Robotics (H.R. Nicholls, ed.), World Scientific, 1992.

Lee, J. C. et al. (2004) „Haptic Pen: Tactile Feedback Stylus for Touch Screens“. Mitsubishi Electric Research Laboratories, Inc., Massachusetts, USA. Retrieved on 06.2006 from: http://www.merl.com/reports/docs/TR2004-133.pdf

Lee, J. C. (2008-1) ”Hacking the Nintendo Wii Remote”, IEEE Pervasive Computing, vol. 7, no. 3, pp. 39-45, Jul-Sept, 2008

Lee, J. C (2008-2) Nintendo Wii projects page. Retrieved on 23.08.2008 from: http://www.cs.cmu.edu/~johnny/projects/wii/

McLaughlin, M., Rizzo, A., Jung, Y., Peng, W., Yeh, S., Zhu, W., and the USC/UT Consortium for Interdisciplinary Research (2005). “Haptics-enhanced virtual environments for stroke rehabilitation.” Proc. IPSI 2005, Cambridge, MA

Mgestyk Technologies Inc. Company homepage accessed and information retrieved on 28.12.2009 from: http://www.mgestyk.com

Microsoft Surface product page. Accessed and information retrieved on 25.01.2010 from: http://www.microsoft.com/surface/Pages/Technical/Learn.aspx

Moore, G. E. (1965) „Gramming more components onto integrated circuit“ Electronics, Volume 38, Number 8, April 19, 1965. Retrieved on 29.05.2009 from: ftp://download.intel.com/museum/Moores_Law/Articles-Press_Releases/Gordon_Moore_1965_Article.pdf /

NationMaster Encyclopaedia „Tactile graphics“ Retrieved on 24.09.2009 from: http://www.nationmaster.com/encyclopedia/Tactile-graphic.

Nintendo 2008 annual report. Downloaded on 29.03.2010 from Nintendo homepage: http://www.nintendo.com/corp/annual_report.jsp

O'Regan, J. K., Noë, A. "A sensorimotor account of vision and visual consciousness." *The Behavioural and brain sciences, vol. 24*, *no. 5*, October 2001.

Ohka, M., Koga, H., Mouri, Y., Sugiura, T., Miyaoka, T., and Mitsuya, Y. (2007) "Figure and texture presentation capabilities of a tactile mouse equipped with a display pad of stimulus pins." *Robotica volume 25*, Issue *4* (pp. 451-460). New York, NY, USA

Oblong Industries Inc. Company homepage accessed and information retrieved on 28.12.2009 from: http://oblong.com/

OpenCV. Open Source Computer Vision Library homepage accessed and information retrieved on 23.08.2008 from: http://www.intel.com/technology/computing/opencv/

Orozco, M., El Saddik, A. (2008) "AdHapticA: Adaptive Haptic Application Framework" IEEE Transactions on Instrumentation and Measurement. Vol 57. Issue: 9, pp 1840 – 1851. ISSN: 0018-9456. Braunschweig, Germany. 2008

Payeur, P., Pasca, C. , Cretu, A.-M. , Petriu, E.M. (2005) "Intelligent Haptic Sensor System for Robotic Manipulation," IEEE Trans. Instrum. Meas., Vol. 54, No. 4, pp. 1583 – 1592, 2005.

Perez, C. A., Holzmann, C. A., Jaeschke, H. E. (2000) „Two-point vibrotactile discrimination related to parameters of pulse burst stimulus." Medical & biological engineering & computing. 2000 Jan;38(1):74-9.

Petriu, E. M., McMath, W. S. , Yeung, S.K., Trif, N., (1992) "Active Tactile Perception of Object Surface Geometric Profiles," IEEE Trans. Instrum. Meas., Vol. 41, No. 1, pp. 87-92, 1992.

Petriu, E. M., McMath, W. S., (1992-2) "Tactile Operator Interface for Semi-autonomous Robotic Applications," Proc.Int. Symposium on Artificial Intell. Robotics Automat. in Space, i-SAIRS'92, pp.77-82, Toulouse, France, 1992

Petriu, E. M., Payeur, P, Cretu, A.-M. (2007) "Haptic Human Interfaces for Robotic Telemanipulation," J. Computing, Vol. 6, Issue 2, pp. 81-88, 2007.

Petriu, E. M., Whalen, T. E., Rudas, I. J., Petriu, D. C., Cordea, M.D. (2008) "Human-Instrument Symbiotic Partnership for Multimodal Environment Perception," Proc. I²MTC 2008 – IEEE Int. Instrum. Meas. Technol. Conf., pp. 1263-1268, Victoria, BC, Canada, May 2008.

Pratt, K. (2001) „Digital Image Processing". Third edition, A Wiley-Interscience Publication, John Wiley & Sons, Inc. New York. ISBN 0-471-22132-5

Regenbrecht, H., Hauber, J., Schoenfelder, R., & Maegerlein, A. (2005). "Virtual reality aided assembly with directional vibro-tactile feedback." In *GRAPHITE '05: Proceedings of the 3rd international conference on Computer graphics and interactive techniques in Australasia and South East Asia* , (pp. 381-387). New York, NY, USA: ACM Press.

Riva, G., David, F., Ijsselsteijn,W. (2003) “Being There: Concepts, Effects and Measurements of User Presence in Synthetic Environments” IOS Press 2003, ISBN: 978-1586033019.

Shapiro, L., and Stockman, G. (2002). „Computer Vision.“ Upper Saddle River, NJ: Prentice Hall.

Sekuler, R. , Balke, R. (1990) „Perception“, 2nd edition, McGraw-Hill, NY, 1990, Chapter 11. Touch, pp.357-383.

Senseg press release (2009) “Senseg brings touch feedback to touch screens and touch panels without mechanical movement” released 03.09.2009. Company homepage accessed and information retrieved on 16.02.2010 from: http://www.senseg.com/wp-content/uploads/0901-senseg-brings-touch-feedback-web.pdf

Seungyon, C. L., Starner, T. (2009) “Mobile gesture interaction using wearable tactile displays.” In *Proceedings of the 27th international conference extended abstracts on Human factors in computing systems,* (pp. 3437-3442). Boston, MA, USA 2009

Shizuki, B., Hisamatsu, T., Takahashi, S., Tanaka, J. (2006) “Laser pointer interaction techniques using peripheral areas of screens”, Proceedings of the working conference on Advanced visual interfaces 2006, pp. 95-98.

Sight and Sound Technology Ltd. Company homepage accessed and information retrieved on 26.04.2010 from: http://www.sightandsound.co.uk/

Smart (2003) Smart technologies „DWiT white paper” Company homepage accessed and document retrieved on 02.06.2006 from: http://www.smarttech.com/dvit/index.asp

Song, J., Haas, C. T., and Caldas, C. H. (2007) „A proximity-based method for locating RFID tagged objects.“ Advanced Engineering Informatics. Vol. 21, Issue 4 Pages 367-376. ISSN: 1474-0346. Elsevier Science Publishers B. V. Amsterdam, The Netherlands, The Netherlands 2007.

Stavens, D. (2008) The OpenCV Library: Computing Optical Flow. Guest Lecture, Computer Science 223b, Stanford University.

Strong, G. W. (1995) „New Directions in Human-Computer Interaction Education, Research and Practice“. Interactions, Vol. 2 , No. 1, pp. 69 – 81.

Vale, P. O., Duncan, N. (2007) “Physical-virtual interpolation”. United States Patent Application 20070188444. Published 16.08.2007. Information retrieved at 17.02.2010 from: http://www.freepatentsonline.com/y2007/0188444.html

Van Dam, A. (1997) „Post-WIMP User Interfaces“ *Communications of the ACM,* 40(2):63-67, 1997

VECIMS 2009. Conference homepage accessed and information retrieved on 24.04.2010 from: http://vecims.ieee-ims.org/2009/index.php

Vicon Motion Systems Inc. Company homepage accessed and information retrieved on 28.12.2009 from: http://www.vicon.com

Watanabe, T., Tamechika, T. and Ifukube, T. (1998) „Exploration of the lines displayed by a tactile mouse“, International Conference on Presentation and Blindness, San Marino, May 1998.

Watanabe, T., Kume, Y. and Ifukube, T. (2000) „Shape discrimination with a tactile mouse“, Journal of the Institute of Image Information and Television Engineers, Vol.54, No.6, pp.840-847, 2000

Wright, M. (2005) “Open Sound Control: an enabling technology for musical networking.” *Org. Sound* 10, 3 (Dec. 2005), pp. 193-200.

Wöldecke, B., Vierjahn, T., Flasko, M., Herder, J., and Geiger, C. (2009) “Steering actors through a virtual set employing vibro-tactile feedback.” In *Proceedings of the 3rd international Conference on Tangible and Embedded interaction* (pp 169-174). New York, NY: ACM

Zhu, S., Yu, A. W., Hawley, D., Roy, R., (1986) “Frustrated total internal reflection: A demonstration and review” in *American Journal of Physics, Volume 54, Issue 7*, pp. 601-607. 07/1986.

Lühikokkuvõte

Käesolev uurimustöö panustab ühte olulisemasse tehnoloogiaga seotud valdkonda: inimese ja masina/arvuti vahelisse kommunikatsiooniprotsessi. Tehnika arenedes tõuseb arvutisüsteemide jõudlus ning samuti kasvab meile igapäevaselt kättesaadava info määr, millega toime tulemiseks vajame me üha efektiivsemaid kasutajaliideseid. Inimese ja arvuti vahelise kasutajaliidese parendamiseks parendamiseks pakume välja järgmiste suhtluskanalite kumulatiivsete rakendamise: mitmesõrme sisend ja puuteaistingul põhinev tagasiside.

Käesolevas uurimistöös esitleme kompeaistingul põhineva tagasiside seadet – korraga mitut (sõrmeotsa) puudet registreeriv kasutajaliides, milles igale sõrmele antakse individuaalne kompeaisting. Lisaks vaatleme ja analüüsime ka mõningaid probleeme, mis sellise kasutajaliidese rakendamisel esile kerkivad. Uurimust eesmärgiks oli töötada välja tehnoloogiline lahendus, mis on lihtsa konstruktsiooniga, töökindel, odav toota ja mida saaks kergesti lisada olemasolevale riistvarale.

Väitekirja põhiline subjekt on uudne puuteaistingul põhineva tagasiside seade ja meetod selle kasutamiseks (seadme ja meetodi uudsust kinnitab Eesti Patendiameti poolt 15.12.2008 väljastatud patent nr: EE05116B1).

Üheks põhjalikumalt vaadeldud probleemiks, mis kaasneb FTIR ja Rear DI tehnoloogiat kasutatavate süsteemidega on CPU suurenenud koormus. See probleem on seda tuntavam, mida suurema resolutsiooniga kaamerat sõrmeotste/puudete positsioneerimiseks kasutatakse. Probleemi olemus seisneb selles, et mida suurem on kaamera ja sellelt pärineva videovoo resolutsioon, seda suurem on ka andmemaht mida tuleb reaalajas töödelda (leida sõrmeotste/puutepunktide asukoht kaadris). Tulemuseks võib kergesti kujuneda olukord, kus kasutajaliides kasutab ära enamiku protsessori võimsusest. Peatükis 4 on kirjeldatud meetodit („Limited Trackframe“ meetod) mis on lihtsalt skaleeritav ja pakub antud probleemile lahendust [Joasoon 2008-2].

Kõrvaluuring on peateemaga seotud järgmiste ühiste punktide kaudu:

- Multipuute tehnoloogia on üheks peamiseks alustehnoloogiaks puuteaistingut tekitava seadme tööks, mistõttu siin pakutud probleemide lahendused sillutavad teed ka peateemas toodud seadme arendamisele.
- Mõlema probleemi lahendus toetub sarnasele lähenemisele (töödeldakse/ kasutatakse ainult seda osa infost, mida hetkel vaja läheb – ökonoomia saavutatakse ebavajalike osade töötlemisest elimineerimise teel)

Lisaks panustab kogu uuring erialasse avades ja sisse juhatades mitmeid uusi võimalikke ja huvitavaid uurimisobjekte nagu näiteks kahetasemelise positsioneerimise kombineerimine „Limited Trakframe“ meetodiga ja kombatava tagasiside kasutamine eraldiseisvates abiseadmetes (mida kirjeldatakse detailsemalt peatükkides „6.1 Contributions“ ja „6.2 Future research“).

Publications by the Author

Joasoon, E.; Mellik, A.; Tulviste, J. (2009). "Employing Haptic Input-Output for Cognitive retraining Applications". In: *VECIMS 2009: Proceeding of: 2009 IEEE International Conference on Virtual Environments, Human-Computer Interfaces and Measurement Systems, Hong Kong, China*, 11.-13.05.2009. IEEE, 2009

Joasoon, E. (2008-1) "Improving the Human-Computer interaction process by means of 2D and 3D visual and haptic interface elements. *Proceedings of 3th IKTDK annual conference publications* (pp.: 151-154) Tallinn: Tallinna Tehnikaülikooli Kirjastus, 2008

Joasoon, E.; Henno, J. (2008-2). A Method to Reduce CPU Overhead in Blob Detection and Tracking Algorithms. In: *Proceedings of 6th IEEE International Conference on Computational Cybernetics*, Stara Lesna, Slovakia. 27.-29.11.2008. IEEE, 2008.

Joasoon, E. (2007). Multi-touch user interface with tactile feedback. In: *Proceedings of Interfaces and Human Computer Interaction 2007: Interfaces and Human Computer Interactions (part of MCCSIS 2007)*. (pp. 223 - 226.) IADIS Press, 2007

Patents by the Author

Joasoon, E. (2008-3). EE 05116 B1 „Meetod ja seade kompimisaistingu vahendusel puutetundliku ekraani kasutajaliideses tagasiside tekitamiseks." (in Estonian) EE Patendileht 4/2008, 15.08.1008. ISSN 1406-0485

APPENDIX A: Patent application.

Joasoon, E. (2008-3). EE 05116 B1 „Meetod ja seade kompimisaistingu vahendusel puutetundliku ekraani kasutajaliideses tagasiside tekitamiseks.“ (in Estonian) Published in: EE Patendileht 4/2008, 15.08.1008. ISSN 1406-0485

EE 05116 B1

(19) EESTI VABARIIK
PATENDIAMET

(11) **EE 05116 B1**

(51) Int.Cl.
G06F 3/00 (2008.04)
G06F 3/01 (2008.04)
G06F 3/033 (2008.04)

(12) **PATENDIKIRJELDUS**

(21) Patenditaotluse number: **P200700001**

(22) Patenditaotluse esitamise kuupäev: **04.01.2007**

(24) Patendi kehtivuse alguse kuupäev: **04.01.2007**

(43) Patenditaotluse avaldamise kuupäev: **15.08.2008**

(45) Patendikirjelduse avaldamise kuupäev: **15.12.2008**

(73) Patendiomanik:

Tallinna Tehnikaülikool
Ehitajate tee 5, 19086 Tallinn, EE

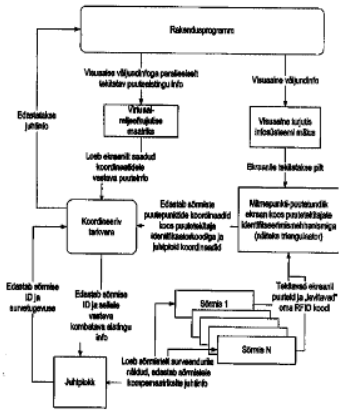
(72) Leiutise autor:

Erkki Joasoon
Sõpruse pst 211-43, 13422 Tallinn, EE

(54) Meetod ja seade kahesuunaliseks puutepõhiseks suhtlemiseks arvuti ja kasutaja vahel

(57) Käesolev leiutis käsitleb meetodit ja seadet kahesuunaliseks puutepõhiseks suhtlemiseks arvuti ja kasutaja vahel, kus on võimaldatud andmete manipuleerimine samaaegselt mitmes ekraanipunktis. Leiutisekohase meetodiga tekitatakse kompimisaisting igale sõrmele individuaalselt ning kompimisaistingu sisu on otseses sõltuvuses iga sõrmeotsa sihist ekraani suhtes.

(57) Present invention relates to a method and device for haptic duplex communication between computer and user, where data manipulation is enabled simultaneously in multiple different screen points. According to the invention the tactile perception is generated for each finger individually and the nature of the tactile perception is in direct dependence of the direction of each fingertip in respect to the screen.



APPENDIX B: Program source code.

C++ program code from “Limited Trackframe” project testing program.

Module daForm is the main user interface object, the window that holds buttons, progress bars, video objects and other visible screen elements.

daForm.h

```
//---------------------------------------------------------------------------
#ifndef daFormH
#define daFormH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ExtCtrls.hpp>
#include <Dialogs.hpp>
#include <ComCtrls.hpp>
//---------------------------------------------------------------------------
class TaForm : public TForm
{
__published:    // IDE-managed Components
        TButton *BT_Load;
        TButton *BT_Track;
        TButton *BT_Capture;
        TButton *BT_Save;
        TButton *BT_Exit;
        TGroupBox *GroupBox1;
        TMemo *SysLog;
```

```
TImage *daImage;
TSaveDialog *aSaveDialog;
TImage *daResultImage;
TGroupBox *GroupBox2;
TLabel *Label1;
TLabel *Label2;
TLabel *Label3;
TLabel *Label4;
TLabel *LB_TimeElapsed;
TLabel *LB_BlobsFound1;
TLabel *LB_Treshold;
TLabel *LB_MinWidth;
TButton *BT_ReportHeader;
TOpenDialog *daOpenDialog;
TButton *Button1;
TGroupBox *GroupBox3;
TTrackBar *TrB_RedTrsh;
TTrackBar *TrB_GreenTrsh;
TTrackBar *TrB_BlueTrsh;
TTrackBar *TrB_minPix;
TLabel *LB_Red;
TLabel *Label6;
TLabel *Label7;
TLabel *LB_Green;
TLabel *Label9;
TLabel *LB_Blue;
TLabel *Label11;
TLabel *LB_MinPix;
TButton *BT_Videostart;
TButton *BT_TrackVideo;
```

```
TButton *BT_StopVideo;
TButton *BT_DefRoi;
TButton *BT_TrackROI;
TButton *BT_VideoAndRoi;
TProgressBar *ProgressBar1;
TLabel *Label5;
TLabel *LB_access_per_sec;
TTimer *Timer1;
TLabel *Label8;
TLabel *LB_10_sec_aver;
TButton *Button2;
void __fastcall BT_ExitClick(TObject *Sender);
void __fastcall FormCreate(TObject *Sender);
void __fastcall BT_CaptureClick(TObject *Sender);
void __fastcall BT_SaveClick(TObject *Sender);
void __fastcall BT_LoadClick(TObject *Sender);
void __fastcall BT_TrackClick(TObject *Sender);
void __fastcall BT_ReportHeaderClick(TObject *Sender);
void __fastcall Button1Click(TObject *Sender);
void __fastcall TrB_RedTrshChange(TObject *Sender);
void __fastcall TrB_GreenTrshChange(TObject *Sender);
void __fastcall TrB_BlueTrshChange(TObject *Sender);
void __fastcall TrB_minPixChange(TObject *Sender);
void __fastcall FormClose(TObject *Sender, TCloseAction &Action);
void __fastcall BT_DefRoiClick(TObject *Sender);
void __fastcall BT_TrackROIClick(TObject *Sender);
void __fastcall BT_VideostartClick(TObject *Sender);
void __fastcall BT_StopVideoClick(TObject *Sender);
void __fastcall BT_TrackVideoClick(TObject *Sender);
void __fastcall BT_VideoAndRoiClick(TObject *Sender);
```

```
        void __fastcall Timer1Timer(TObject *Sender);
        void __fastcall Button2Click(TObject *Sender);
private: // User declarations
public:         // User declarations
        __fastcall TaForm(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TaForm *aForm;
//---------------------------------------------------------------------------
#endif
```

daForm.cpp

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "time.h"
#include "cvcam.h"
#include "cxcore.h"
#include "highgui.h"
#include "Graphics.hpp"
#include "btStructDef.h"
#include "myFuncs.h"
#include "ej_BlobTracker.h"
#include "btFunctions.h"
#include "daForm.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
```

```
#pragma resource "*.dfm"

TaForm *aForm;
// Global settings structure
_btGSettings *btGlobalSettings = new _btGSettings;
// BlobTrack settings structure
_btSettings *BlobTrackSettings=new _btSettings;
// Frame, that we gonna blobtrack and a buffer and the image to paint the results!
IplImage *frame2Process, *retriiver, *trackResults;
// some var to calculate execution time
clock_t startT, endT, elapsedT;
_btStatCount *Statistics = new _btStatCount;
_btTrackResults *TrackResultVector=new _btTrackResults;
int tenSecCounter=0;
double tenSecCycles=0;


//---------------------------------------------------------------------------
__fastcall TaForm::TaForm(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TaForm::BT_ExitClick(TObject *Sender)
{
        //some carbage collect here ...
  //    delete btGlobalSettings;
  //    delete BlobTrackSettings;
  //    delete Statistics;
        aForm->Close();
```

```
}
//---------------------------------------------------------------------------
void __fastcall TaForm::FormCreate(TObject *Sender)
{
        // clear system log window and write programm started.
        myClearSysLog(SysLog);
        myAddLog("Program started",SysLog);
        // give default values to global settings structure.
        btInitGlobalSettings(btGlobalSettings,SysLog);
        // give default values to BlobTrack settings structure.
        btInitBlobTrackSettings(BlobTrackSettings,SysLog);
        // create headers and allocate data
        frame2Process=cvCreateImage(cvSize(640,480) , IPL_DEPTH_8U, 3);
        retriiver=cvCreateImage(cvSize(640,480) , IPL_DEPTH_8U, 3);
        trackResults=cvCreateImage(cvSize(640,480) , IPL_DEPTH_8U, 3);

}
//---------------------------------------------------------------------------
void __fastcall TaForm::BT_CaptureClick(TObject *Sender)
{
        CvCapture *aCapture;
        aCapture=cvCreateCameraCapture(btGlobalSettings-
>SelectedCam_Prim);
        if (cvGrabFrame(aCapture )){
                retriiver=cvRetrieveFrame(aCapture);
                cvCopy(retriiver,frame2Process);
                ej_Ipl2VCLImage(frame2Process, daImage);
                btGlobalSettings->StillTaken=1;
                myReportIplParam(frame2Process,SysLog);
                }
                else
```

```
                {myAddLog("Nothing to capture!!", SysLog);}
                cvReleaseCapture(&aCapture);
}
//---------------------------------------------------------------------------


void __fastcall TaForm::BT_SaveClick(TObject *Sender)
{
        if (btGlobalSettings->StillTaken) {
                aSaveDialog->DefaultExt="bmp";
                aSaveDialog->Filter="*.bmp";
                aSaveDialog->InitialDir=".";
                if(aSaveDialog->Execute()){
                                myAddLog(aSaveDialog->FileName,SysLog);
                                daImage->Picture->SaveToFile(aSaveDialog-
>FileName);
                                //cvSaveImage(aSaveDialog->FileName.c_str(),
image2Process);
                        }
                        else{
                                myAddLog("filename not selected",SysLog);}
                }
                else{
                                myAddLog("Frame not captured, nothing to save",
SysLog);}
}
//---------------------------------------------------------------------------

void __fastcall TaForm::BT_LoadClick(TObject *Sender)
{
        daOpenDialog->DefaultExt="bmp";
```

```
        daOpenDialog->Filter="*.bmp";
        daOpenDialog->InitialDir=".";
        if(daOpenDialog->Execute()){
                                myAddLog(daOpenDialog->FileName,SysLog);
                                retriiver=cvLoadImage(daOpenDialog-
>FileName.c_str(),CV_LOAD_IMAGE_ANYCOLOR);
                                frame2Process=cvCloneImage(retriiver);
                                //cvFlip(retriiver, frame2Process,0);
                                cvCopy(retriiver, frame2Process,0);
                                cvNamedWindow("Capture",1);
                                cvShowImage("Capture",frame2Process);
                                //cvReleaseImage(&retriiver);
                                //ej_Ipl2VCLImage(frame2Process, daImage);
                        }
                        else{
                                myAddLog("filename not selected",SysLog);}
}


//---------------------------------------------------------------------------

void __fastcall TaForm::BT_TrackClick(TObject *Sender)
{

        if (frame2Process) {
                trackResults=cvCloneImage(frame2Process);
                btZeroStats(Statistics);
                TrackResultVector->nBlobs=0;
                startT=clock();
                        for (int i=0; i < 25; i++) btShowBlobs(frame2Process,
trackResults, BlobTrackSettings, TrackResultVector,0);
```

```
                endT=clock();
                elapsedT=endT-startT;
                LB_TimeElapsed->Caption=elapsedT;
                cvNamedWindow("Tracked",1);
                cvShowImage("Tracked",trackResults);
                cvCopy(trackResults, frame2Process);
                //ej_Ipl2VCLImage(trackResults,daResultImage);
                btReportStats(Statistics,SysLog);
                btReportBlobTrackStats(TrackResultVector,SysLog);
                btDoCircles(daResultImage, TrackResultVector);
                cvReleaseImage(&trackResults);
                }
        else{
                myAddLog("Nothing to process!!",SysLog);}


}
//---------------------------------------------------------------------------

void __fastcall TaForm::BT_ReportHeaderClick(TObject *Sender)
{
        myAddLog("*** Frame2Process ***",SysLog);
        myReportIplParam(frame2Process,SysLog);
        myAddLog("*** Retriiver ***",SysLog);
        myReportIplParam(retriiver,SysLog);
        myAddLog("*** trackResults ***",SysLog);
        myReportIplParam(trackResults,SysLog);
}
//---------------------------------------------------------------------------
```

```
void __fastcall TaForm::Button1Click(TObject *Sender)
{
                myClearSysLog(SysLog);
}
//---------------------------------------------------------------------------

void __fastcall TaForm::TrB_RedTrshChange(TObject *Sender)
{
        BlobTrackSettings->ColorTresh.Red=(unsigned char)TrB_RedTrsh->Position;
        LB_Red->Caption=TrB_RedTrsh->Position;
        //btUpdateBlobTrackSettings(TrB_RedTrsh->Position, &BlobTrackSettings->ColorTresh.Red,LB_Red);
}
//---------------------------------------------------------------------------

void __fastcall TaForm::TrB_GreenTrshChange(TObject *Sender)
{
        BlobTrackSettings->ColorTresh.Green=(unsigned char)TrB_GreenTrsh->Position;
        LB_Green->Caption=TrB_GreenTrsh->Position;
        //btUpdateBlobTrackSettings(TrB_GreenTrsh->Position, &BlobTrackSettings->ColorTresh.Green,LB_Green);
}
//---------------------------------------------------------------------------

void __fastcall TaForm::TrB_BlueTrshChange(TObject *Sender)
{
        BlobTrackSettings->ColorTresh.Red=(unsigned char)TrB_BlueTrsh->Position;
        LB_Blue->Caption=TrB_BlueTrsh->Position;
```

```
        //btUpdateBlobTrackSettings(TrB_BlueTrsh->Position,
&BlobTrackSettings->ColorTresh.Blue,LB_Blue);
}
//---------------------------------------------------------------------------

void __fastcall TaForm::TrB_minPixChange(TObject *Sender)
{
        BlobTrackSettings->minWidth=TrB_minPix->Position;
        LB_MinPix->Caption=TrB_minPix->Position;
}
//---------------------------------------------------------------------------

void __fastcall TaForm::FormClose(TObject *Sender, TCloseAction &Action)
{
        //some carbage collect here ...
        if (btGlobalSettings->VideoFeedActive) {
                cvcamStop();
                cvcamExit();
        }
        cvDestroyAllWindows();
        delete btGlobalSettings;
        delete BlobTrackSettings;
        delete Statistics;
        btDestructTrackResults(TrackResultVector);
        delete TrackResultVector;

}
//---------------------------------------------------------------------------


void __fastcall TaForm::BT_DefRoiClick(TObject *Sender)
```

```
{
        if (TrackResultVector->nBlobs) {

        btCalcTrackFrame(TrackResultVector,BlobTrackSettings,frame2Process,1
00);
                        cvRectangle(frame2Process, cvPoint( BlobTrackSettings-
>TrackFrame.X,  BlobTrackSettings->TrackFrame.Y ), cvPoint(
BlobTrackSettings->TrackFrame.X+BlobTrackSettings->TrackFrame.deltaX,
BlobTrackSettings->TrackFrame.Y+BlobTrackSettings->TrackFrame.deltaY ),
cvScalar(255,0,0),2,8,0);
                        cvNamedWindow("Frame limiter",1);
                        cvShowImage("Frame limiter",frame2Process);
//                      daResultImage->Canvas->Pen->Color=clBlue;
//                      daResultImage->Canvas->Brush->Style=bsClear;
//                      daResultImage->Canvas->Rectangle(BlobTrackSettings-
>TrackFrame.X,287-BlobTrackSettings->TrackFrame.Y,BlobTrackSettings-
>TrackFrame.X+BlobTrackSettings->TrackFrame.deltaX,287-(BlobTrackSettings-
>TrackFrame.Y+BlobTrackSettings->TrackFrame.deltaY));
                }
                else {
                myAddLog("No blobs detected, cannot calculate
trackframe",SysLog);
      }
}
//---------------------------------------------------------------------------

void __fastcall TaForm::BT_TrackROIClick(TObject *Sender)
{

        if (frame2Process) {
                trackResults=cvCloneImage(frame2Process);
                btZeroStats(Statistics);
                TrackResultVector->nBlobs=0;
```

```
                startT=clock();
                        for (int i=0; i < 25; i++) btShowBlobs(frame2Process,
trackResults, BlobTrackSettings, TrackResultVector,1);
                endT=clock();
                elapsedT=endT-startT;
                LB_TimeElapsed->Caption=elapsedT;
                ej_Ipl2VCLImage(trackResults,daResultImage);
                btReportStats(Statistics,SysLog);
                btReportBlobTrackStats(TrackResultVector,SysLog);
                btDoCircles(daResultImage, TrackResultVector);
                cvReleaseImage(&trackResults);
                }
        else{
                myAddLog("Nothing to process!!",SysLog);}


}
//---------------------------------------------------------------------------

void __fastcall TaForm::BT_VideostartClick(TObject *Sender)
{
        cvNamedWindow("CamOut",1);
        cvcamWindow camOutWin =(cvcamWindow)
cvGetWindowHandle("CamOut");
        cvcamSetProperty(0, CVCAM_PROP_ENABLE, CVCAMTRUE);
        cvcamSetProperty(0, CVCAM_PROP_RENDER, CVCAMTRUE);
        cvcamSetProperty(0, CVCAM_PROP_CALLBACK, btCallback);
        cvcamSetProperty(0, CVCAM_PROP_WINDOW, &camOutWin);
        cvcamInit();
        cvcamStart();
        btGlobalSettings->VideoFeedActive=0;
```

```
        myAddLog("Camera started!!", SysLog);
}
//---------------------------------------------------------------------------

void __fastcall TaForm::BT_StopVideoClick(TObject *Sender)
{
        cvcamStop();
}
//---------------------------------------------------------------------------

void __fastcall TaForm::BT_TrackVideoClick(TObject *Sender)
{
        cvNamedWindow("TrackRes",1);
        btGlobalSettings->BlobTrackVideoActive=1;
}
//---------------------------------------------------------------------------

void __fastcall TaForm::BT_VideoAndRoiClick(TObject *Sender)
{
//      cvNamedWindow("TrackRes",1);
//      btGlobalSettings->BlobTrackVideoActive=1;
        btGlobalSettings->BlobTrackVideoROIActive=1;
}
//---------------------------------------------------------------------------

void __fastcall TaForm::Timer1Timer(TObject *Sender)
{
        // call to function display last second stats.
        // reset the stats.
        double tempCycl=0;
```

```
        tempCycl=(Statistics->PixelReads+Statistics->PixelWrites+Statistics-
>ArrayLoops)/1000;
        LB_access_per_sec->Caption=tempCycl;
        Statistics->PixelReads=0;
        Statistics->PixelWrites=0;
        Statistics->ArrayLoops=0;
        tenSecCounter++;
        tenSecCycles+=tempCycl;
        if (tenSecCounter==10) {
                tenSecCounter=0;
                LB_10_sec_aver->Caption=tenSecCycles/10;
                tenSecCycles=0;
        }
}
//---------------------------------------------------------------------------


void __fastcall TaForm::Button2Click(TObject *Sender)
{
        cvcamSetProperty(0, CVCAM_PROP_ENABLE, CVCAMTRUE);
        cvcamSetProperty(0, CVCAM_PROP_RENDER, CVCAMTRUE);
        cvcamGetProperty(0, CVCAM_VIDEOFORMAT, NULL);
}
//---------------------------------------------------------------------------
```

Module MyFuncs contains functions that perform various supporting tasks (painting circles on displayed videostream, writing things to log files etc.)

MyFuncs.h

```
//---------------------------------------------------------------------------

#ifndef MyFuncsH
#define MyFuncsH
//---------------------------------------------------------------------------
void myAddLog(AnsiString aText,TMemo *aMemo);
void myAddLogNCR(AnsiString aText,TMemo *aMemo);
void myClearSysLog(TMemo *aMemo);
void myReportIplParam(IplImage *image,TMemo *SysLog);
Graphics::TBitmap* ej_Ipl2Bitmap(IplImage *image);
void ej_Ipl2VCLImage(IplImage *aImage, TImage *vclImage);
void myDoCircle(TImage *vclImage,int CentreX,int CentreY,int aRadius);
void myDoCircle(IplImage *image,int CentreX,int CentreY, int aRadius);

#endif
```

MyFuncs.cpp

```
//---------------------------------------------------------------------------



#pragma hdrstop


//#include "cvcam.h"
//#include "cv.h"
//#include "highgui.h"
#include "cxcore.h"
#include "btStructDef.h"
#include "daForm.h"
#include "Graphics.hpp"
#include "MyFuncs.h"
```

```
//---------------------------------------------------------------------------
#pragma package(smart_init)

// A function to add text to System Log Window
void myAddLog(AnsiString aText,TMemo *aMemo){
        aMemo->Text=aMemo->Text+aText+"\r\n";
}

// A function to add text to System Log Window with No Carriage Return
void myAddLogNCR(AnsiString aText,TMemo *aMemo){
        aMemo->Text=aMemo->Text+aText;
}

// A Function to clear System Log Window.
void myClearSysLog(TMemo *aMemo){
        aMemo->Text="";
}

void myReportIplParam(IplImage *image,TMemo *SysLog){
        if (image) {
                        // image initialised ... gen report
                myAddLogNCR("Image dimensions W:" ,SysLog);
                myAddLogNCR(image->width,SysLog);
                myAddLogNCR(" , H:",SysLog);
                myAddLog(image->height,SysLog);
                myAddLogNCR("Color Model: " ,SysLog);
                myAddLog(image->colorModel,SysLog);
                myAddLogNCR("Channel seq: ",SysLog);
                myAddLog(image->channelSeq,SysLog);
                myAddLogNCR("Channels: " ,SysLog);
```

```
                myAddLog(image->nChannels,SysLog);
                myAddLogNCR("Depth: ",SysLog);
                myAddLog(image->depth,SysLog);
                myAddLogNCR("nSize: ",SysLog);
                myAddLog(image->nSize,SysLog);
                myAddLogNCR("Image size: ",SysLog);
                myAddLog(image->imageSize,SysLog);
                }
                else {
                myAddLog("Image not initialised!",SysLog);}
}

// A Function to convert IplImage to bitmap.
Graphics::TBitmap* ej_Ipl2Bitmap(IplImage *image){

        TColor tempPixel;
        tempPixel = 0;
        int tempInt;
        int tempR,tempG,tempB;
        unsigned char* tempIndex;
        tempIndex = NULL;

        Graphics::TBitmap* returnBitmap;
        returnBitmap = new Graphics::TBitmap();
        returnBitmap->Width = image->width;
        returnBitmap->Height = image->height;

        for(int k = 0;k < image->width; ++k)
        {
                for(int l = 0;l < image->height; ++l)
```

```
                {
                        tempInt = 0;
                        tempIndex = &((unsigned char*)(image->imageData + image->widthStep*l))[k*3];
                        tempB = tempIndex[0];
                        tempG = tempIndex[1];
                        tempR = tempIndex[2];

                        tempB <<= 16;
                        tempG <<= 8;
                        tempInt = tempR + tempB + tempG;

                        tempPixel = tempInt;
                        returnBitmap->Canvas->Pixels[k][image->height-l]        = tempPixel;
                }
        }
        return(returnBitmap);
}

void ej_Ipl2VCLImage(IplImage *aImage, TImage *vclImage){
                Graphics::TBitmap* aBitmap=ej_Ipl2Bitmap(aImage);
                vclImage->Picture->Bitmap = aBitmap;
                aBitmap->FreeImage();
                delete aBitmap;
}

void myDoCircle(TImage *vclImage,int CentreX,int CentreY, int aRadius){
        vclImage->Canvas->Pen->Width=2;
        vclImage->Canvas->Pen->Color=clGreen;
        vclImage->Canvas->Brush->Style=bsClear;
```

```
        vclImage->Canvas->Ellipse(CentreX-aRadius,CentreY-
aRadius,CentreX+aRadius,CentreY+aRadius);
}

void myDoCircle(IplImage *image,int CentreX,int CentreY, int aRadius){
        cvCircle( image,cvPoint( CentreX, CentreY), aRadius,cvScalar( 0,
255,0),1,8,0 );
}
```

Module btFunctions contains functions that support blobtracking and integrates it with IO screen.

btFunctions.h

```
//---------------------------------------------------------------------------
#ifndef btFunctionsH
#define btFunctionsH
//---------------------------------------------------------------------------
void btInitGlobalSettings(_btGSettings *gSettings,TMemo *aMemo);
void btInitBlobTrackSettings(_btSettings *BlobTrackSettings,TMemo *aMemo);
void btZeroStats(_btStatCount *Statistics);
void btReportStats(_btStatCount *Statistics,TMemo *aMemo);
void btUpdateBlobTrackSettings(int newValue, unsigned char *btSetting,TLabel
*aLabel);
void btReportBlobTrackStats(_btTrackResults *TrackResultVector,TMemo
*aMemo);
void btDoCircles(TImage *vclImage,_btTrackResults *TrackResultVector);
void btDoCircles(IplImage *image,_btTrackResults *TrackResultVector);
void btCallback(IplImage *Image);
#endif
```

btFunctions.cpp

```
//---------------------------------------------------------------------------


#pragma hdrstop

#include "cvcam.h"
#include "cxcore.h"
#include "highgui.h"
#include "btStructDef.h"
#include "daForm.h"
#include "MyFuncs.h"
#include "ej_BlobTracker.h"
```

```
#include "btFunctions.h"

//---------------------------------------------------------------------------
#pragma package(smart_init)

extern IplImage *frame2Process, *retriiver, *trackResults;
extern _btSettings *BlobTrackSettings;
extern _btTrackResults *TrackResultVector;
extern _btStatCount *Statistics;
extern _btGSettings *btGlobalSettings;

void btInitGlobalSettings(_btGSettings *gSettings,TMemo *aMemo){
        // set yet unused parameters to default values.
        gSettings->SelectedCam_Prim=-1; //Primary camera not selected
        gSettings->CamsAvailble=0;              //No cameras found yet
        gSettings->StillTaken=0;       //Stillimage not taken
        gSettings->BlobTreshold=50;       //blobtracking treshold;
        gSettings->MinHorBlobSize=2;//Minimal amount of hor. pixels that make
up a blob
        gSettings->VideoFeedActive=0;   //Video input not activated.
        gSettings->BlobTrackVideoActive=0;
        gSettings->BlobTrackVideoROIActive=0;


        //Get the amount of avalible cameras connected.
        gSettings->CamsAvailble=cvcamGetCamerasCount();
        // Report the found number of cameras to Syslog.
        myAddLogNCR(gSettings->CamsAvailble, aMemo);
        myAddLog(" connected cameras found.", aMemo);
        // If cameras found, select cam0 as primary camera.
        if (gSettings->CamsAvailble>0) {
                gSettings->SelectedCam_Prim=0;
                CameraDescription *aCamDescr=new CameraDescription;
                cvcamGetProperty(0, CVCAM_DESCRIPTION, aCamDescr);
                myAddLogNCR("Selecting "", aMemo);
                myAddLogNCR(aCamDescr->DeviceDescription,aMemo);
                myAddLog("" as Primary camera by default", aMemo);
                delete aCamDescr;
        }

        // update Syslog with setup completed message
        myAddLog("Global Settings Structure initialised.", aMemo);

};
```

```
void btInitBlobTrackSettings(_btSettings *BlobTrackSettings,TMemo *aMemo){
	BlobTrackSettings->treshold=250;
	BlobTrackSettings->minWidth=5;
	BlobTrackSettings->ColorTresh.Red=150;
	BlobTrackSettings->ColorTresh.Blue=150;
	BlobTrackSettings->ColorTresh.Green=150;
	BlobTrackSettings->TrackFrame.active=0;
}

void btZeroStats(_btStatCount *Statistics){
	Statistics->BlobsFound1Pass=0;
	Statistics->BlobsFound2Pass=0;
	Statistics->PixelReads=0;
	Statistics->PixelWrites=0;
}

void btReportStats(_btStatCount *Statistics,TMemo *aMemo){
		myAddLogNCR("Blobs found 1. pass: " ,aMemo);
		myAddLog(Statistics->BlobsFound1Pass, aMemo);
		myAddLogNCR("Blobs found 2. pass: " ,aMemo);
		myAddLog(Statistics->BlobsFound2Pass, aMemo);
		myAddLogNCR("Pixelreads: " ,aMemo);
		myAddLog(Statistics->PixelReads, aMemo);
		myAddLogNCR("Pixelwrites: " ,aMemo);
		myAddLog(Statistics->PixelWrites, aMemo);
		myAddLogNCR("Arrayloops: " ,aMemo);
		myAddLog(Statistics->ArrayLoops, aMemo);

}
void btReportBlobTrackStats(_btTrackResults *TrackResultVector,TMemo
*aMemo){
	if (TrackResultVector->nBlobs>0) {
		for (int i = 0; i < TrackResultVector->nBlobs; i++) {
			myAddLogNCR("Blob " ,aMemo);
			myAddLog((i+1) ,aMemo);
			myAddLogNCR("Witdh: " ,aMemo);
			myAddLog(TrackResultVector->BlobData[i]->Width,
aMemo);
			myAddLogNCR("Height: " ,aMemo);
			myAddLog(TrackResultVector->BlobData[i]->Height,
aMemo);
			myAddLogNCR("X: " ,aMemo);
```

```
                        myAddLog(TrackResultVector->BlobData[i]->CentreX,
aMemo);
                        myAddLogNCR("Y: " ,aMemo);
                        myAddLog(TrackResultVector->BlobData[i]->CentreY,
aMemo);
                        myAddLogNCR("Pixels: " ,aMemo);
                        myAddLog(TrackResultVector->BlobData[i]->nPixels,
aMemo);
                }
        }
};

void btUpdateBlobTrackSettings(int newValue, unsigned char *btSetting,TLabel
*aLabel){
        btSetting=(unsigned char*)newValue;
        aLabel->Caption=newValue;
}

void btDoCircles(TImage *vclImage,_btTrackResults *TrackResultVector){
                for (int i = 0; i < TrackResultVector->nBlobs; i++) {
                        myDoCircle(vclImage,TrackResultVector->BlobData[i]-
>CentreX,287-TrackResultVector->BlobData[i]->CentreY,5);
                }
}

void btDoCircles(IplImage *image,_btTrackResults *TrackResultVector){
                for (int i = 0; i < TrackResultVector->nBlobs; i++) {
                        myDoCircle(image,TrackResultVector->BlobData[i]-
>CentreX,TrackResultVector->BlobData[i]->CentreY,5);
                }
}

void btCallback(IplImage *image){
        if (btGlobalSettings->BlobTrackVideoActive) {
//                      IplImage *aImage=new IplImage;
                        IplImage *bImage=new IplImage;
//                      aImage=cvCloneImage(image);
                        bImage=cvCloneImage(image);
                        btZeroStats(Statistics);
                        TrackResultVector->nBlobs=0;
                        btShowBlobs(image, bImage, BlobTrackSettings,
TrackResultVector,btGlobalSettings->BlobTrackVideoROIActive);
                        btDoCircles(bImage,TrackResultVector);
                        if (btGlobalSettings->BlobTrackVideoROIActive) {
```

```
        btCalcTrackFrame(TrackResultVector,BlobTrackSettings,image,100);
                        cvRectangle(bImage, cvPoint( BlobTrackSettings-
>TrackFrame.X, BlobTrackSettings->TrackFrame.Y ), cvPoint(
BlobTrackSettings->TrackFrame.X+BlobTrackSettings->TrackFrame.deltaX,
BlobTrackSettings->TrackFrame.Y+BlobTrackSettings->TrackFrame.deltaY ),
cvScalar(255,0,0),2,8,0);
                }
                cvShowImage("TrackRes", bImage);
//              cvReleaseImage(&aImage);
                cvReleaseImage(&bImage);
        }

}
```

Module ej_BlobTracker contains functions that handle blobtracking, "Limited Trackframe" method implementation and performance measuring.

Ej_BlobTracker.h

```
//---------------------------------------------------------------------------

#ifndef ej_BlobTrackerH
#define ej_BlobTrackerH
//---------------------------------------------------------------------------
#endif

struct _btIndex{
        int FirstPass;
        int SecondPass;
};

struct _btScanIndex{
        int height;
        int width;
        int **Index;
};

struct _btCoord{
        int nCol;
        int nRow;
};

struct _btPixTreshold{
        unsigned char Red;
        unsigned char Blue;
        unsigned char Green;
};

struct _btPixel{
        unsigned char Red;
        unsigned char Blue;
        unsigned char Green;
        _btCoord Address;
};

struct _btTempBlobData{
        int nBlobIndex;
        int minX;
```

```
        int maxX;
        int minY;
        int maxY;
};

struct _aBlobData{
        int CentreX;
        int CentreY;
        int Width;
        int Height;
        int nPixels;
        _btTempBlobData tmpData;
};

struct _btROI{
        int active;
        int rebuild;
        int X;
        int Y;
        int deltaX;
        int deltaY;
};

struct _btTrackResults{
        int nBlobs;
        _aBlobData **BlobData;
};

struct _btSettings{
        unsigned char treshold;
        _btPixTreshold ColorTresh;
        int minWidth;
        _btROI TrackFrame;
};

void btShowBlobs(IplImage *aImage,IplImage *bImage, _btSettings
*settings,_btTrackResults *TrackResultVector,int trackFrame);
void btShowBlobsVideo(IplImage *aImage,IplImage *bImage, _btSettings
*settings,_btTrackResults *TrackResultVector,int trackFrame);
int btInitMatrix(_btScanIndex *aMatrix);
void btDestructMatrix(_btScanIndex *aMatrix);
void btInitIPLMatrix(_btScanIndex *aMatrix, IplImage *image);
void btScanBlobs1Pass(IplImage *image,_btScanIndex *aMatrix,_btIndex
*BlobCounter,_btSettings *settings,int trackFrame);
```

```
void btScanBlobs2Pass(_btScanIndex *aMatrix,_btIndex *blobCounter,_btSettings
*settings,int trackFrame);
void btCheckBlobStart(int nCol, int nRow, IplImage *image,_btScanIndex
*aMatrix,_btIndex *blobCounter,_btSettings *settings);
void btCheckBlobContinue(int nCol, int nRow, IplImage *image,_btScanIndex
*aMatrix,_btSettings *settings);
int btCheckNPixels(int nCol, int nRow, IplImage *image,int nPixels,_btSettings
*settings);
void btWritePixel(IplImage *image,int x,int y,_btPixel *pixel);
void btReadPixel(IplImage *image,int x,int y,_btPixel *pixel);
void btShow1PassResults(IplImage *bImage,_btScanIndex *aMatrix);
void btCreateResultArray(_btScanIndex *aMatrix,_btIndex
*blobCounter,_btSettings *settings, _btTrackResults *TrackResultVector,int
trackFrame);
void btCreateResultArrayVideo(_btScanIndex *aMatrix,_btIndex
*blobCounter,_btSettings *settings,_btTrackResults *TrackResultVector,int
trackFrame);
void btInitTrackResults(_btTrackResults *TrackResultVector);
void btDestructTrackResults(_btTrackResults *TrackResultVector);
void btCalcTrackFrame(_btTrackResults *TrackResultVector,_btSettings
*BlobTrackSettings,IplImage *frame2Process,float overlay);
```

Ej_BlobTracker.cpp

```
//---------------------------------------------------------------------------
#pragma hdrstop

#include "cxcore.h"
// ***** TESTROWS!! Remove later!!!!!***********
#include "daForm.h"
#include "MyFuncs.h"
#include "btStructDef.h"
// ********* EOTR *****************************
#include "ej_BlobTracker.h"

//---------------------------------------------------------------------------

#pragma package(smart_init)

// ***** TESTROWS!! Remove later!!!!!***********
extern _btStatCount *Statistics;
// ********* EOTR *****************************
```

```
// function that tracks blobs from first image and paints them to second one.
void btShowBlobs(IplImage *aImage,IplImage *bImage, _btSettings
*settings,_btTrackResults *TrackResultVector,int trackFrame){
//          cvFlip(aImage,bImage,0);
//          cvSetZero(bImage);
            _btIndex *blobCounter=new _btIndex;
            blobCounter->FirstPass=0;
            blobCounter->SecondPass=0;
            // define matrix for first pass scancodes.
            _btScanIndex *ScanIndex=new _btScanIndex;
            // initialize matrix.
            btInitIPLMatrix(ScanIndex, aImage);
            // scan image for blobs 1. pass
            btScanBlobs1Pass(aImage,ScanIndex,blobCounter,settings,trackFrame);
            btScanBlobs2Pass(ScanIndex,blobCounter,settings,trackFrame);

            // ***** TESTROWS!! Remove later!!!!!***********
            // Report blobcount
            Statistics->BlobsFound1Pass=blobCounter->FirstPass;
            Statistics->BlobsFound2Pass=blobCounter->SecondPass;
            // ********* EOTR ****************************


            // now its time 2 check!!
            btShow1PassResults(bImage, ScanIndex);
            btCreateResultArray(ScanIndex,blobCounter,settings,TrackResultVector,tr
ackFrame);

            delete blobCounter;
            btDestructMatrix(ScanIndex);
}


void btScanBlobs2Pass(_btScanIndex *aMatrix,_btIndex *blobCounter,_btSettings
*settings,int trackFrame){
            if (blobCounter->FirstPass==0) return;
            int nCol,nRow;
            int startCol,endCol,startRow,endRow;
            // check for trackframe usage
            if (trackFrame==0 || settings->TrackFrame.active==0||settings-
>TrackFrame.rebuild) {
                        startCol=0;
                        endCol=aMatrix->width;
                        startRow=0;
```

```
                endRow=aMatrix->height;}
        else{
                startCol=settings->TrackFrame.X;
                endCol=settings->TrackFrame.X+settings-
>TrackFrame.deltaX;
                startRow=settings->TrackFrame.Y;
                endRow=settings->TrackFrame.Y+settings-
>TrackFrame.deltaY;}
        //initialize blob vector

        // ***** TESTROWS!! Remove later!!!!!***********
//      myAddLog("**SecondPass****: ",aForm->SysLog);
//      myAddLogNCR("startCol: ",aForm->SysLog);
//      myAddLog(startCol,aForm->SysLog);
//      myAddLogNCR("endCol: ",aForm->SysLog);
//      myAddLog(endCol,aForm->SysLog);
//      myAddLogNCR("startRow: ",aForm->SysLog);
//      myAddLog(startRow,aForm->SysLog);
//      myAddLogNCR("endRow: ",aForm->SysLog);
//      myAddLog(endRow,aForm->SysLog);
        // ********* EOTR ****************************

        int *blobVector= new int[blobCounter->FirstPass+1];
        // reset blob vector elements
        for (int i = 0; i < blobCounter->FirstPass+1; i++){
         blobVector[i]=0;}

        for (nCol = startCol; nCol < endCol; nCol++) {
                for (nRow = startRow; nRow < (endRow-1); nRow++) {
                        // Are we in the blob area?
                        if (aMatrix->Index[nCol][nRow]!=0){
                                // we are in the blob area
                                // is there a blob area down, that we can merge
                                if (aMatrix->Index[nCol][nRow+1]!=0){
                                        //is it allready merged?
                                        if (blobVector[aMatrix-
>Index[nCol][nRow+1]]!=0){
                                                // it is allready merged with
something,
                                                int upVal=blobVector[aMatrix-
>Index[nCol][nRow]];
                                                int downVal=blobVector[aMatrix-
>Index[nCol][nRow+1]];
```

```
                                                if (upVal==downVal){// Blobs are
allready merged... do nothing!
                                                }
                                                else {
                                                        if (upVal<downVal &&
upVal!=0) { //replace all downwal's with upVals.
                                                                for (int i=0; i <
aMatrix->Index[nCol][nRow+1]; i++) {
                                                                        if
(blobVector[i]==downVal) blobVector[i]=upVal;
                                                                }
                                                        }
                                                        else{//replace all upVal's
with downVals.
                                                                if (upVal==0) {

        blobVector[aMatrix->Index[nCol][nRow]]=blobVector[aMatrix-
>Index[nCol][nRow+1]];
                                                                }
                                                                else {
                                                                        for (int
i=0; i < aMatrix->Index[nCol][nRow+1]; i++) {
                                                                                if
(blobVector[i]==upVal) blobVector[i]=downVal;
                                                                        }
                                                                }
                                                        }
                                                }
                                        }
                                        else {// it is not merged, mark for merging.
                                                // is current area merged allready??
                                                if (blobVector[aMatrix-
>Index[nCol][nRow]]) {// merged, assigning merge index
                                                        blobVector[aMatrix-
>Index[nCol][nRow+1]]=blobVector[aMatrix->Index[nCol][nRow]];
                                                }
                                                else { // not merged, assigning new
index to both!
                                                        blobCounter-
>SecondPass++;
                                                        blobVector[aMatrix-
>Index[nCol][nRow]]=blobCounter->SecondPass;
                                                        blobVector[aMatrix-
>Index[nCol][nRow+1]]=blobCounter->SecondPass;
```

```
                                                }
                                        }
                                }
                        }
                        Statistics->ArrayLoops++;
                } // end nRow cycle
        } // end nCol cycle
        // updating blob code matrix
        for (nCol = startCol; nCol < endCol; nCol++) {
                for (nRow = startRow; nRow < endRow; nRow++) {
                        aMatrix->Index[nCol][nRow]=blobVector[aMatrix-
>Index[nCol][nRow]];
                        Statistics->ArrayLoops++;
                }
        }
        delete [] blobVector;
}

void btScanBlobs1Pass(IplImage *image,_btScanIndex *aMatrix,_btIndex
*blobCounter,_btSettings *settings,int trackFrame){
        int nCol,nRow;
        int startCol,endCol,startRow,endRow;
        // check for trackframe usage
        if (trackFrame==0 || settings->TrackFrame.active==0||settings-
>TrackFrame.rebuild) {
                        startCol=0;
                        endCol=image->width;
                        startRow=0;
                        endRow=image->height;}
                else{
                        startCol=settings->TrackFrame.X;
                        endCol=settings->TrackFrame.X+settings-
>TrackFrame.deltaX;
                        startRow=settings->TrackFrame.Y;
                        endRow=settings->TrackFrame.Y+settings-
>TrackFrame.deltaY;}
        // Loop through rows..

        // ***** TESTROWS!! Remove later!!!!!**********
//      myAddLog("**FirstPass****: ",aForm->SysLog);
//      myAddLogNCR("startCol: ",aForm->SysLog);
//      myAddLog(startCol,aForm->SysLog);
//      myAddLogNCR("endCol: ",aForm->SysLog);
//      myAddLog(endCol,aForm->SysLog);
```

```
//          myAddLogNCR("startRow: ",aForm->SysLog);
//          myAddLog(startRow,aForm->SysLog);
//          myAddLogNCR("endRow: ",aForm->SysLog);
//          myAddLog(endRow,aForm->SysLog);
          // ********* EOTR ****************************

          for (nRow = startRow; nRow < endRow; nRow++) {
                    // loop trough pixels in the row
                    for (nCol = startCol; nCol < (endCol-settings->minWidth);
nCol++) {
                              if (aMatrix->Index[nCol][nRow]==0){
                                        // Not a blob area, lets see if we can mark one!
                                        btCheckBlobStart(nCol, nRow, image, aMatrix,
blobCounter, settings);
                              }
                              else{
                                        // we are in a blob area, lets see if we can skip to
next pixel
                                        if (aMatrix->Index[nCol+1][nRow]==0){
                                                  if (aMatrix->width<=(nCol+1+settings-
>minWidth)) {
                                                            // we are in the end of blob area
and in the end of picture, we should check all remining pixels
                                                            int reminingPix = aMatrix->width
- (nCol+1);
                                                            for (int i = 0; i < reminingPix; i++)
{
                                                                      if
(btCheckNPixels(nCol+i,nRow,image,1,settings)==0) break;
                                                            }
                                                  }
                                                  else {
                                                            // we are in the end of blob area,
but in the middle of picture, lets see if we can extend the blob
                                                            btCheckBlobContinue(nCol+1,
nRow,image, aMatrix, settings);}
                                        }
                                        else{
                                                  // we are in the middle of marked blob,
skip to next pixel.
                                        }//end else
                              }//end else
                    }//end for nCol
          }//end for nRow
```

```
}//end of function

void btCheckBlobStart(int nCol, int nRow, IplImage *image,_btScanIndex
*aMatrix,_btIndex *blobCounter,_btSettings *settings){
        if (btCheckNPixels(nCol,nRow,image,settings->minWidth,settings)){
                //new blob area confirmed
                blobCounter->FirstPass++;
                for (int i = 0; i < settings->minWidth; i++) {
                        if (nCol+i>(aMatrix->width-1)) nCol=aMatrix->width-
(i+1);
                        aMatrix->Index[nCol+i][nRow]=blobCounter->FirstPass;
                        Statistics->ArrayLoops++;
                }
         }
}

void btCheckBlobContinue(int nCol, int nRow, IplImage *image,_btScanIndex
*aMatrix,_btSettings *settings){
        if (nCol-1<0) return; //something wrong with the index
        if (btCheckNPixels(nCol,nRow,image,1,settings)){
                //next pixel passed test
                aMatrix->Index[nCol][nRow]=aMatrix->Index[nCol-1][nRow];
        //      return 1;
        }
        // return 0;
}

int btCheckNPixels(int nCol, int nRow, IplImage *image,int nPixels,_btSettings
*settings){
        // check for improper nCol and nRow values
        if (nRow>image->height) return 0;
        if (nCol>image->width-nPixels) return 0;

        _btPixel *aPixel=new _btPixel;
        for (int i = 0; i < nPixels; i++) {
          btReadPixel(image,nCol+i,nRow,aPixel);
          if (aPixel->Red<settings->ColorTresh.Red||aPixel->Blue<settings-
>ColorTresh.Blue||aPixel->Green<settings->ColorTresh.Green) {
                        // check failed!
                        delete aPixel;
                        return 0;}
        }// endloop, if we are still here the check succseeded!
        delete aPixel;
        return 1;
```

```
}

void btWritePixel(IplImage *image,int x,int y,_btPixel *pixel){
	int i;
	if (x>image->width) x=image->width;
	if (y>image->height) y=image->height;
	i= y*image->widthStep+x*3;
	image->imageData[i]=pixel->Blue;
	image->imageData[i+1]=pixel->Green;
	image->imageData[i+2]=pixel->Red;
	// ***** TESTROWS!! Remove later!!!!!**********
	Statistics->PixelWrites++;
	// ********* EOTR *****************************

}

void btReadPixel(IplImage *image,int x,int y,_btPixel *pixel){
	int i;
	if (x>image->width) x=image->width;
	if (y>image->height) y=image->height;
	i= y*image->widthStep+x*3;
	pixel->Blue=image->imageData[i];
	pixel->Green=image->imageData[i+1];
	pixel->Red=image->imageData[i+2];
	// ***** TESTROWS!! Remove later!!!!!**********
	Statistics->PixelReads++;
	// ********* EOTR *****************************

}


void btInitIPLMatrix(_btScanIndex *aMatrix, IplImage *image){
	aMatrix->height=image->height;
	aMatrix->width=image->width;
	btInitMatrix(aMatrix);
}

int btInitMatrix(_btScanIndex *aMatrix){
	int nCol,nRow;
	// Check if matrix size is valid.
	if (aMatrix->height==0) return 0;
	if (aMatrix->width==0) return 0;
	// Allocate memory for the matrix
	aMatrix->Index= new int *[aMatrix->width+1];
```

```
	for (nCol = 0; nCol < (aMatrix->width+1); nCol++) {
		aMatrix->Index[nCol]=new int [aMatrix->height+1];}
	// Fill matrix with zeros
	for (nCol=0; nCol < (aMatrix->width+1); nCol++) {
		for (nRow=0; nRow < (aMatrix->height+1); nRow++){
			aMatrix->Index[nCol][nRow]=0;}
			//Statistics->ArrayLoops++;
	}
	return 1;
}

void btDestructMatrix(_btScanIndex *aMatrix){
	int nCol;
	if (aMatrix->height){
		for( nCol = 0 ; nCol < (aMatrix->width+1) ; nCol++ )
			delete [] aMatrix->Index[nCol] ;
		delete [] aMatrix->Index ;}
		delete aMatrix;
};

void btShow1PassResults(IplImage *bImage,_btScanIndex *aMatrix){
	// First we blacken the output image
	// cvSetZero(bImage);
	int nCol, nRow;
	_btPixel *redPixel=new _btPixel;
	redPixel->Red=255;
	redPixel->Blue=0;
	redPixel->Green=0;
	for (nCol = 0; nCol < aMatrix->width; nCol++) {
		for (nRow = 0; nRow < aMatrix->height; nRow++) {
			if (aMatrix->Index[nCol][nRow])
	btWritePixel(bImage,nCol,nRow,redPixel);
			//Statistics->ArrayLoops++;
		}
	}
	delete redPixel;
}

void btCreateResultArray(_btScanIndex *aMatrix,_btIndex
*blobCounter,_btSettings *settings,_btTrackResults *TrackResultVector,int
trackFrame){
	//1. Check if settings allready have a structure (old) and delete it!
	//2. Create new empty array for blobs settings
	//3. create a temp array
```

```
	//4. Fill temp array!!
	//5. Recalculate temp array into settings for return.
	int nCol, nRow, bIndex;
	int startCol,endCol,startRow,endRow;
	// check for trackframe usage
	if (trackFrame==0 || settings->TrackFrame.active==0||settings-
>TrackFrame.rebuild) {
			startCol=0;
			endCol=aMatrix->width;
			startRow=0;
			endRow=aMatrix->height;}
		else{
			startCol=settings->TrackFrame.X;
			endCol=settings->TrackFrame.X+settings-
>TrackFrame.deltaX;
			startRow=settings->TrackFrame.Y;
			endRow=settings->TrackFrame.Y+settings-
>TrackFrame.deltaY;}
	//initialize blob vector

	// ***** TESTROWS!! Remove later!!!!!***********
//	myAddLog("**Result array generation****: ",aForm->SysLog);
//	myAddLogNCR("startCol: ",aForm->SysLog);
//	myAddLog(startCol,aForm->SysLog);
//	myAddLogNCR("endCol: ",aForm->SysLog);
//	myAddLog(endCol,aForm->SysLog);
//	myAddLogNCR("startRow: ",aForm->SysLog);
//	myAddLog(startRow,aForm->SysLog);
//	myAddLogNCR("endRow: ",aForm->SysLog);
//	myAddLog(endRow,aForm->SysLog);
	// ********* EOTR ****************************

	btDestructTrackResults(TrackResultVector);
	TrackResultVector->nBlobs=blobCounter->SecondPass;
	btInitTrackResults(TrackResultVector);
	for (nCol = startCol; nCol < endCol; nCol++) {
		for (nRow = startRow; nRow < endRow; nRow++) {
			bIndex=aMatrix->Index[nCol][nRow];
			// if (bIndex > blobCounter->SecondPass) {bIndex=0;}
			if (bIndex > 0) {
				if (TrackResultVector->BlobData[bIndex-1]-
>tmpData.minX==-1||TrackResultVector->BlobData[bIndex-1]-
>tmpData.minX>nCol){TrackResultVector->BlobData[bIndex-1]-
>tmpData.minX=nCol;}
```

```
                                if (TrackResultVector->BlobData[bIndex-1]-
>tmpData.minY==-1||TrackResultVector->BlobData[bIndex-1]-
>tmpData.minY>nRow){TrackResultVector->BlobData[bIndex-1]-
>tmpData.minY=nRow;}
                                if (TrackResultVector->BlobData[bIndex-1]-
>tmpData.maxX==-1||TrackResultVector->BlobData[bIndex-1]-
>tmpData.maxX<nCol){TrackResultVector->BlobData[bIndex-1]-
>tmpData.maxX=nCol;}
                                if (TrackResultVector->BlobData[bIndex-1]-
>tmpData.maxY==-1||TrackResultVector->BlobData[bIndex-1]-
>tmpData.maxY<nRow){TrackResultVector->BlobData[bIndex-1]-
>tmpData.maxY=nRow;}
                                TrackResultVector->BlobData[bIndex-1]-
>nPixels++;
                                TrackResultVector->BlobData[bIndex-1]-
>tmpData.nBlobIndex=bIndex;
                        }
                        Statistics->ArrayLoops++;
                }
        }
        for (int i = 0; i < blobCounter->SecondPass; i++) {
                if (TrackResultVector->BlobData[i]->tmpData.nBlobIndex!=i+1){

//                      myAddLog("Blobcounter integrity fault",aForm-
>SysLog);
//                      myAddLogNCR("Blobcounter value: ",aForm->SysLog);
//                      myAddLog((i+1),aForm->SysLog);
//                      myAddLogNCR("TrackResultVector Blobindex : ",aForm-
>SysLog);
//                      myAddLog(TrackResultVector->BlobData[i]-
>tmpData.nBlobIndex,aForm->SysLog);


                        } // peaks messima, et data integrity fault;
                TrackResultVector->BlobData[i]->Width=TrackResultVector-
>BlobData[i]->tmpData.maxX-TrackResultVector->BlobData[i]->tmpData.minX;
                TrackResultVector->BlobData[i]->Height=TrackResultVector-
>BlobData[i]->tmpData.maxY-TrackResultVector->BlobData[i]->tmpData.minY;
                TrackResultVector->BlobData[i]->CentreX=TrackResultVector-
>BlobData[i]->tmpData.minX+(TrackResultVector->BlobData[i]->Width/2);
                TrackResultVector->BlobData[i]->CentreY=TrackResultVector-
>BlobData[i]->tmpData.minY+(TrackResultVector->BlobData[i]->Height/2);
                Statistics->ArrayLoops++;
        }
```

```
}

void btInitTrackResults(_btTrackResults *TrackResultVector){
	TrackResultVector->BlobData=new _aBlobData*[TrackResultVector-
>nBlobs];
	for (int i = 0; i < (TrackResultVector->nBlobs); i++) {
		TrackResultVector->BlobData[i]=new _aBlobData;
		TrackResultVector->BlobData[i]->CentreX=0;
		TrackResultVector->BlobData[i]->CentreY=0;
		TrackResultVector->BlobData[i]->Width=0;
		TrackResultVector->BlobData[i]->Height=0;
		TrackResultVector->BlobData[i]->nPixels=0;
		TrackResultVector->BlobData[i]->tmpData.minX=-1;
		TrackResultVector->BlobData[i]->tmpData.maxX=-1;
		TrackResultVector->BlobData[i]->tmpData.minY=-1;
		TrackResultVector->BlobData[i]->tmpData.maxY=-1;
		TrackResultVector->BlobData[i]->tmpData.nBlobIndex=0;
	}
}

void btDestructTrackResults(_btTrackResults *TrackResultVector){
	if (TrackResultVector->nBlobs>0){
		for (int i=0; i < TrackResultVector->nBlobs; i++){
			delete TrackResultVector->BlobData[i];}
		TrackResultVector->nBlobs=0;
		delete [] TrackResultVector->BlobData;
		}
}

void btCalcTrackFrame(_btTrackResults *TrackResultVector,_btSettings
*BlobTrackSettings,IplImage *aImage,float overlay){
	int Xmin=-1;
	int Xmax=-1;
	int Ymin=-1;
	int Ymax=-1;
	int tWidth,tHeight;
	float xOverlay,yOverlay;
	for (int i = 0; i < TrackResultVector->nBlobs; i++) {
		if (Xmin==-1||Xmin>TrackResultVector->BlobData[i]-
>CentreX){Xmin=TrackResultVector->BlobData[i]->CentreX;}
		if (Xmax==-1||Xmax<TrackResultVector->BlobData[i]-
>CentreX){Xmax=TrackResultVector->BlobData[i]->CentreX;}
		if (Ymin==-1||Ymin>TrackResultVector->BlobData[i]-
>CentreY){Ymin=TrackResultVector->BlobData[i]->CentreY;}
```

```
                if (Ymax==-1||Ymax<TrackResultVector->BlobData[i]-
>CentreY){Ymax=TrackResultVector->BlobData[i]->CentreY;}
                }
        tWidth=Xmax-Xmin;
        tHeight=Ymax-Ymin;
        xOverlay=tWidth*(overlay/200);

        // ***** TESTROWS!! Remove later!!!!!**********
//      myAddLogNCR("xOverlay: ",aForm->SysLog);
//      myAddLog(xOverlay,aForm->SysLog);
        // ********* EOTR *****************************

        yOverlay=tHeight*(overlay/200);

        // ***** TESTROWS!! Remove later!!!!!**********
//      myAddLogNCR("yOverlay: ",aForm->SysLog);
//      myAddLog(yOverlay,aForm->SysLog);
        // ********* EOTR *****************************

        if ((Xmin-xOverlay)<0) {Xmin=0;}
                else {Xmin=Xmin-xOverlay;}
        if ((Xmax+xOverlay)>(aImage->width-1)) {Xmax=aImage->width-1;}
                else {Xmax=Xmax+xOverlay;}
        if ((Ymin-yOverlay)<0) {Ymin=0;}
                else {Ymin=Ymin-yOverlay;}
        if ((Ymax+yOverlay)>(aImage->height-1)) {Ymax=aImage->height-1;}
                else {Ymax=Ymax+yOverlay;}
        BlobTrackSettings->TrackFrame.active=1;
        if (TrackResultVector->nBlobs<2){ BlobTrackSettings-
>TrackFrame.rebuild=1;}
                else {BlobTrackSettings->TrackFrame.rebuild=0;}
        BlobTrackSettings->TrackFrame.X=Xmin;
        BlobTrackSettings->TrackFrame.Y=Ymin;
        BlobTrackSettings->TrackFrame.deltaX=Xmax-Xmin;
        BlobTrackSettings->TrackFrame.deltaY=Ymax-Ymin;
}
```

APPENDIX C: Curriculum Vitae (in Estonian)

1. Isikuandmed

   Ees- ja perekonnanimi: Erkki Joasoon
   Sünniaeg ja -koht: 01.02.1968, Tallinn
   Kodakondsus: Eesti

2. Kontaktandmed

   Aadress: Raja 15, 12618 Tallinn
   Telefon: +3725138331
   E-posti aadress: erkkijoasoon@hot.ee

3. Hariduskäik

Õppeasutus (nimetus lõpetamise ajal)	**Lõpetamise aeg**	**Haridus (eriala/kraad)**
Tallinna Tehnikaülikool	1993	Inseneridiplom (elektriinsener automaatika erialal) ekviv. M.Sc.ga
Tallinna 37. Keskkool	1986	Keskhariduse diplom

4. Keelteoskus (alg-, kesk- või kõrgtase)

Keel	**Tase**
Eesti keel	Kõrgtase
Inglise keel	Kõrgtase
Vene keel	Kesktase
Soome keel	Kõrgtase

5. Teenistuskäik

Töötamise aeg	Tööandja nimetus	Ametikoht
Alates 2008	Unlimited Media OÜ	Tegevjuht / Omanik
2008 - 2008	Linxtelecom Eesti	Operatsioonide juht
2006 - 2008	PKC Grupp Eesti	IT Juht
2005 - 2006	Finantsinspektsioon	IT Audiitor
2002 - 2004	Stockmann Eesti	IT Juht
2000 - 2002	Hansapank Markets	IT Haldusjuht
1999 – 2000	Ericsson Eesti	Mobiilsidevõrkude tehniline tugi
1997 - 1998	Mars Inc	Infrastructure Service Centre Analyst
1994 - 1997	Masterfoods Baltic	Kohalike IT süsteemide tehniline tugi
1992 - 1994	AS Aetec	Programmeerija

6. Teadustegevus

Joasoon, E.; Mellik, A.; Tulviste, J. (2009). "Employing Haptic Input-Output for Cognitive retraining Applications." In: VECIMS 2009: Proceeding of: 2009 IEEE International Conference on Virtual Environments, Human-Computer Interfaces and Measurement Systems, Hong Kong, China, 11.-13.05.2009. IEEE, 2009

Joasoon, E.; Henno, J. (2008). "A Method to Reduce CPU Overhead in Blob Detection and Tracking Algorithms". In: Proceedings of 6th IEEE International Conference on Computational Cybernetics, Stara Lesna, Slovakia. 27.-29.11.2008. IEEE, 2008.

Joasoon, Erkki (2008). "Inimese ja infosüsteemi vahelise kommunikatsiooni parendamine kahe ja kolmemõõtmeliste visuaalsete ja haptiliste liideste abil." Info- ja kommunikatsioonitehnoloogia doktorikooli IKTDK kolmanda

aastakonverentsi artiklite kogumik (lk: 151-154) Tallinn: Tallinna Tehnikaülikooli Kirjastus, 2008,

Joasoon, E. (2007). "Multi-touch user interface with tactile feedback." In: Proceedings of Interfaces and Human Computer Interaction 2007: Interfaces and Human Computer Interactions (part of MCCSIS 2007) . (pp. 223 - 226.) IADIS Press, 2007

Joasoon, E. „Meetod ja seade kompimisaistingu vahendusel puutetundliku ekraani kasutajaliideses tagasiside tekitamiseks", EE Patendileht 4/2008, 15.08.1008. ISSN 1406-0485.

7. Kaitstud lõputööd

   „Müügimehe töökoht"

   Inseneridiplom (ekviv. M.Sc.) Tallinna Tehnikaülikool, 1993

8. Teadustöö põhisuunad

   Inimese ja arvuti vaheline kommunikatsioon, kombatav-tunnetuslik tagasiside, objektide analüüs videostriimis.

APPENDIX D: Curriculum Vitae

1. Personal data

Name: Erkki Joasoon
Date and place of birth: 01.02.1968, Tallinn
Citizenship: Estonian

2. Contact information

Address: Raja 15, 12618 Tallinn
Phone: +3725138331
E-mail: erkkijoasoon@hot.ee

3. Education

Educational institution	**Graduation year**	**Education (field of study/degree)**
Tallinn University of Technology	1993	Engineering (Electrical engineer on a field of control engineering) eq. M.Sc.
Tallinn 37. Secondary School	1986	Secondary education

4. Language skills (basic-,intermediate- or high level)

Language	**Level**
Estonian	High level
English	High level
Russian	Intermediate
Finnish	High level

5. Professional Employment

Period	Employer	Position
Since 2008	Unlimited Media OÜ	CEO/Owner
2008 - 2008	Linxtelecom Estonia	Operations manager
2006 - 2008	PKC Group Estonia	IT Manager
2005 - 2006	Financial Supervision Authorities	IT Auditor
2002 - 2004	Stockmann OÜ	IT manager
2000 - 2002	Hansapank Markets	IT Maintenance manager
1999 – 2000	Ericsson Estonia	Technical support on mobile networks
1997 - 1998	ISI/Mars Inc	Infrastructure Service Centre Analyst
1994 - 1997	Masterfoods Baltic	IT Local System Management Support
1992 - 1994	AS Aetec	Programmer

6. Scientific work

Joasoon, E. “Multi-touch user interface with tactile feedback” pg. 223-226 In: VECIMS 2009: Proceeding of: 2009 IEEE International Conference on Virtual Environments, Human-Computer Interfaces and Measurement Systems, Hong Kong, China, 11.-13.05.2009. IEEE, 2009

Joasoon, E. “Improving the Human-Computer interaction process by means of 2D and 3D visual and haptic interface elements.” Proceedings of 3th IKTDK annual conference publications (pp.: 151-154) Tallinn: Tallinna Tehnikaülikooli Kirjastus, 2008

Joasoon, E. Henno, J. “A method to reduce CPU overhead in blob detection and tracking algorithms.” In: Proceedings of 6th IEEE International Conference on Computational Cybernetics, Stara Lesna, Slovakia. 27.-29.11.2008. IEEE, 2008.

Joasoon, E. „Meetod ja seade kompimisaistingu vahendusel puutetundliku ekraani kasutajaliideses tagasiside tekitamiseks.“ (in Estonian) EE Patendileht 4/2008, 15.08.1008. ISSN 1406-0485.

Joasoon, E. , Mellik, A. , Tulviste, J. (2009). “Employing Haptic Input-Output for Cognitive retraining Applications.” In: VECIMS 2009: Proceeding of: 2009 IEEE International Conference on Virtual Environments, Human-Computer Interfaces and Measurement Systems, Hong Kong, China, 11.-13.05.2009. IEEE, 2009

7. Defended theses

   „Computer application for salesman workstation“

   Dipl. engineering (eq. M.Sc.) Tallinn University of Technology, 1993

8. Research interests

   Human computer interaction, tactile feedback, object tracking and recognition in video stream.