demeTouch
a New Approach to Human Music Interaction

Department of Electronic Systems
Aalborg University

June 6, 2007

Alexandre Fleury

AALBORG UNIVERSITY
DEPARTMENT OF ELECTRONIC SYSTEMS

Niels Jernes Vej 12 ▪ DK-9220 Aalborg Ø Phone +45 9635 8650

TITLE: demeTouch: a New Approach to Human Music Interaction
THEME: Intelligent Multimedia
PROJECT PERIOD: 10th Semester, February 2007 - June 2007
PROJECT GROUP: 1072a

PARTICIPANT
Alexandre Fleury

SUPERVISOR
Zheng-Hua Tan

Abstract

This report documents the work of Alexandre Fleury through the 10th semester of the Intelligent Multimedia master program at Aalborg University's Department of Electronic Systems.
The main objective of this project is to create a finger controlled music application for manipulating an audio library based on a music-color association.
Two main phases are necessary to reach this goal: first, to study the constraints imposed by the table-top display used as a support, in order to develop adaptive and robust finger detection methods. This finger detector has to be fast enough to process the video stream in live and implement a solution to handle the calibration issue introduced by the physical setup. Secondly, to create a graphical user interface that considers these constraints and assume that the user is new to associate music with colors. The navigation through the application as well as the color-based playlist creation has to be intuitive.
This document reports the preanalysis, analysis, design, implementation and testing parts of the project.

PUBLICATIONS: 3
NUMBER OF PAGES: 111
FINISHED: 6th June 2007

Preface

This report is written during the project period of the 10^{th} semester at the Department of Electronic Systems, Aalborg University.

Details concerning the report structure and style

The report is built on the following structure: first, the *Introduction* chapter presents the general context of the project. Secondly, the *Preanalysis* chapter provides the reader some background needed to understand the project, by detailing the previous projects that demeTouch is related to. Then the *Analysis* chapter gives the project both a guideline by setting up a main goal to aim at, and explains the technological material (the tools) used to reach this goal. Later the *Design* chapter explains and justifies the way these tools are used to achieve the project's goal, while the *Implementation* chapter details the challenges of realizing the solutions previously designed. Finally, the *Tests* chapter studies the performances of the developed system, before the *Conclusion* gives an objective view upon the achieved system.

A user of the system described in the report is always referred in text as "she", while developers and test users are referred as "he". The author of the report is referred as "the author".

Collaborative work

In agreement with the supervisor and the E-Study Board, this project has been conducted partly in collaboration with Frédéric Boulet, student in the Intelligent Multimedia master program at Aalborg University. The reader can refer to [Bou07] for the details concerning his work. The reason for this collaboration is a common interest in the original project proposed by Kasper Løvborg Jensen. The common work consists in the researches about the reacTable project, the tabletop display setup (including the choice of the camera) and the development of the finger recognition algorithms. In the present report, it concerns section 3.5 in the analysis chapter and section 4.2 in the design chapter.

Acknowledgments

The author would like to sincerely thank everyone who directly or indirectly helped me conduct his master's thesis.

First, the staff of Aalborg University Electronic System Department, for their contribution to the project in terms of project management and supervision.

- Zheng-Hua Tan, who supervised the project, providing guidelines and ideas for the technical part, giving useful feedback for the report and keeping an eye on the time schedule in order to respect deadlines,
- Charlotte Skindbjerg Pedersen, the Intelligent Multimedia master program's secretary, who took care of all the administrative work concerning the project,
- Kasper Løvborg Jensen, PhD student in the Department of Electronic System, who proposed the project's original idea, and provided useful feedback throughout the semester,
- Lars Bo Larsen, associate professor at the Department of Electronic Systems, coordinator of the Intelligent Multimedia master program, who kept an attentive eye on the project from a higher perspective.

Concerning the project design and implementation, the author would like to thank

- Frédéric Boulet, for the collaboration in defining the table's setup and designing the finger detection algorithms,

- Martin Kaltenbrunner and the reacTable's staff, who kindly gave answers about the physical setup and the video processing engine,

- Guillaume Jacquemin and the demelo's staff, who expressed their interest in demeTouch by providing relevant and full of insight advice about the project.

The author would like to express his special gratitude to Sara, who patiently supported his work in the successful moments as well as in the more difficult times. Finally, many thanks to Emilien, Julien and Pierre-André, who created a lively atmosphere around the project.

Aalborg University, June 2007

Alexandre Fleury

Contents

CHAPTER 1

Introduction

This chapter introduces the project from a general point of view. It sets up the main issues involved by the techniques used without detailing them. It also presents the delimitations fixed to the project, in order to stay on course. These "boundaries" avoid the natural tendency to keep on spreading the original idea of the project, while the deadline imposes to reach a certain level of achievement in due time.

1.1 Project definition

The basic idea of demeTouch is to offer the user a new approach to her music library by exciting three of her senses:

- **Sight** when using the music-color association, and navigate through the application designed in harmony with the "color" of the music
- **Touch** when interacting with the music library and creating playlists using fingers on the table's surface
- **Hearing** when listening to a piece of music or a playlist according to her mood

Thus, she will be able to navigate through her audio library using fingers in contact with a table's surface, select a piece of music according to her mood and play it. She will be also able to create playlists by choosing the sequence of colors according to her mood which the music played will be automatically chosen.

To implement such a system, four main components will be used: first, the tabletop display developed by students of the Aalborg University Department of Electronic Systems (Aalborg, Denmark), which will be improved by adding a video processing engine to distinguish fingers in physical contact with the table. This engine will be inspired by the work of the reacTable project, realized by a team from the Universitat Pompeu Fabra (Barcelona, Spain). The music application will use the music-color associator developed by a team of students from the Ecole Centrale d'Electronique (Paris, France). Finally, the application layer of the system will be developed from scratch to cope with the issues introduced by the tabletop display and to offer the user an intuitive and pleasant interface.

1.2 Main issues

This project relies on the combination of the four independent elements previously introduced: the physical setup, the back-end video processing engine, the color-music association processor and the front-end application. Except from the color-music association processor, all of them present challenges as they has to be adapted before being integrated into the new system. In addition, the combination of the four elements itself creates new challenges in terms of optimization of time processing or calibration. The next paragraphs introduce these issues. They will be discussed in details in chapter 4, where solutions will be proposed.

The physical setup The tabletop display is made of two separate elements: the external structure and a removable mirror. They have to be correctly associated to offer the possibility of using a projector and a camera in an optimized manner.

The video processing engine Recognizing objects (fingers) from a live video stream requires two characteristics: robustness and speed. The algorithm has to take into account the noise embedded in the images given by the camera and be fast enough with low latency to provide a quick response to the user's interactions with the table.

The color-music association processor This system will be reuse without modification. Every time it will be called, it will take a piece of music as an input and give a set of features as an output. The integration of this processor will have to be smooth making its use fast. Later, this component will be associated to the music application.

The music application Users will interact with the system via a piece of software dedicated to managing a record library. Such an application presents several constraints in term of intuitiveness and visual rendering. Manipulating the application has to respect basic conventions to give the user a familiar feeling especially that the way of interacting with the system is unconventional.

Other issues Merging the tabletop display with the finger recognition engine will be the most delicate combination task: the layout of the physical elements will determine the accuracy of the video processing engine. A bad scheme of arrangement could lead to very poor results in term of recognition, thus the video processing engine will require to be calibrated before usage to ensure the highest recognition rate. Another example of physical adjustments concerns the ambient light: as the camera has an autoadaptative

response to the light it receives, the table will benefit from a brown out instead of a strong lightning system. Concerning the application, its layout will have to take into account the table surface design and the net area to display the visual elements. Finally, the graphical elements' shape, size and location will be deducted from the recognition accuracy and the zones where object detection gives better or worse results.

1.3 Delimitations

This project has a demonstration purpose: instead of revolutionizing the field of video processing systems, it aims to illustrate the possibilities given by the existing tabletop display. The core idea behind this project is to combine existing techniques to provide a new approach to multimedia content manipulation. The difficulty mainly comes from the time period allocated to the project, during which the three systems previously introduced have to be assimilated, modified and combined with together.

Due to this limited time, the first priority of this project will be to have an up running application with basic working functions. To do so, the focus will be first put on the video processing engine, in order to have an acceptable finger detector engine. Then the application will be developed step by step, from a simple color based music player aiming to a fully integrated audio library manager.

CHAPTER 2

Preanalysis

This chapter focuses on the theoretical background of the project. It introduces the three separate projects that demeTouch either reuse or is inspired by: the tabletop display, the reacTable and demelo. The contribution of each of them is explained.

2.1 The existing material: the tabletop display

2.1.1 AAU's tabletop display

The tabletop available at the Aalborg University's Department of Electronic Systems has been originally created by Søren Larsen to study the use of tabletop displays (see [Lar05]). The original setup was composed of an external structure, a mirror and a projector, as shown in Figure 2.1. Users interacted with the setup via ultrasound pens that provide their (x, y) coordinates on the table's surface.

Figure 2.1: The first version of the tabletop, by Søren Larsen

Then, a group of students from the Intelligent Multimedia program reused the setup to explore the field of 3D interactions on a tabletop display [JKR06]. Then, Christian Oelholm, Jesper Graarup Jensen and Simon Kofod built another version of the tabletop display in a smoother and built-in manner: the structure was entirely made of wood, the projector and mirror being hidden below the table, as depicted in Figure 2.2.

2.1.2 Applications developed for the setup

The aim of this setup is to provide a base for studies in the following fields: interactions with tabletop displays, collaborative work, educational and entertaining applications.

Figure 2.2: The second version of the tabletop, by Christian Oelholm, Jesper Graarup Jensen and Simon Kofod

The first version of the tabletop display has been used to create a game where a single user had to draw his way through rotating labyrinths in a limited time. Then, the second version were used to create a 3D environment that was updated depending on the user location around the table, to offer the proper perspective and feeling of depth. Finally, a third application has been developed using the tabletop display as a support to study collaborative work, via a puzzle creation game, where several users had to collaborate or compete in solving puzzles, using the concept of public / private areas.

2.1.3 The tabletop display, demeTouch's physical setup

Tabletop displays offer an innovative support for displaying and manipulating virtual content. It also introduces new challenges due to the dimension of the display area, and the interactions between the numerical data and the user. By using such a physical support, demeTouch offers the user a new approach to her digital media library, usually manipulated via a classical mouse / keyboard input set.

2.2 The original project: reacTable

2.2.1 A music instrument using tangibles

Figure 2.3: Illustration of the reacTable's concept

reacTable is an innovative project in the field of music creation. It offers the user the possibility to create pieces of music using tangibles and fingers. These objects are placed on the surface of a back lighted table, their position and motion being analyzed to influence the musical creation. [KJGA06] introduces this original concept, developed by a team of "digital luthiers" from Barcelona, Spain. Several videos are available on the project's website[1].

2.2.2 How does it work?

The table

As depicted in Figure 2.4, the design of the table itself is original and innovative. Information are exchanged by the table's surface and the computer vision engine in a bidirectional manner:

- the **objects laying on the table** (tangibles and fingers) are filmed: their position, orientation and motion are recorded. These information are used to interpret the musician's actions and act according to them.

[1] `http://mtg.upf.edu/reactable/?media`

- the **graphical environment** is projected on the table's surface, providing feedback on the objects' action.

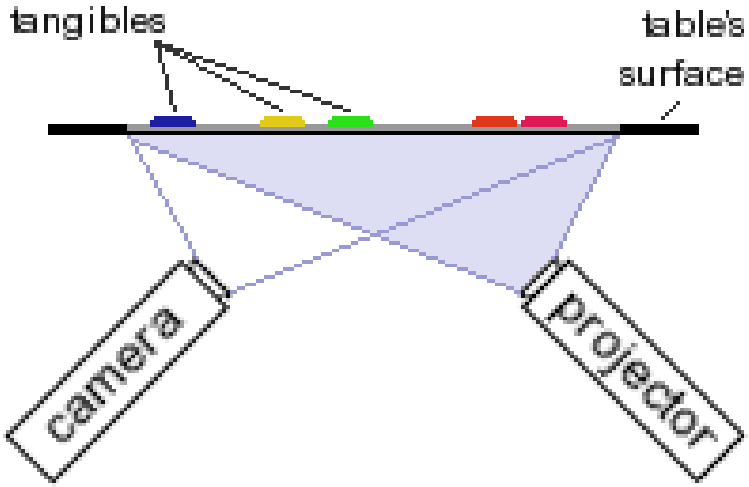


Figure 2.4: The table design (simplified): information exchanged between the table's surface and the computer vision engine are bidirectional

If the basic idea is rather simple, its application needs to address several computer vision issues, described in [KB07]. These can be summed up in the two following questions. Although they are introduced here, these issues are detailed later in chapter 4 when discussing about setting up the table used for demeTouch.

How to get a proper view of the table? First, the viewing angle (most cameras' lens capture with a narrow one) can be enlarged by using a mirror. Then, if graphical information are to be projected on the table, ones should light the table with an infrared source and use a visible light blocker on the camera to clearly film the objects laying on the table's surface.

How to recognize and distinguish the tangibles and fingers? The reacTable team introduced a robust solution to the recognition and tracking problems: objects are associated with fiducials, easily distinguishable from one to another. Once recognized, the fiducial's structure indicates its position and orientation, and events are associated with the apparition, disappearance and moves recorded. Fingers are considered as other objects, fiducials being associated to fingerprints.

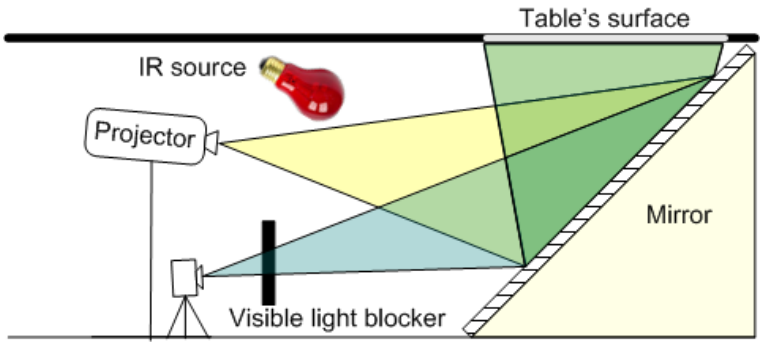


Figure 2.5: The table design (real), including a mirror, infrared sources and visible light blocker

The musical application

The application developed for the table is an instrument used to create digital pieces of music using tangibles and fingers. The core idea described in [KJGA06] is that each tangible laying on the table takes part in the musical creation, by either directly producing sound or modifying the existing musical piece. The user can interfere in the objects' layout on the table by adding and removing some, moving them closer or away... By assembling tangibles together, she can modify the original effect of the object to create new ones. Demonstrations of this instrument are available on the project's website.

2.2.3 Related projects

The reacTable system offers an infinite range of usage, only limited by imagination. A selection of accomplished projects can be consulted from the project's website. Various ideas have already been implemented onto the original reacTable project, some of them being very convincing.

MUSICtable

Amoung these projects, the closest one to demeTouch is the MUSICtable. [SGVF05] introduces the concept of this project which consists in a map-based representation of the user's audio library, from which is created playlist according to her mood. By choosing a starting point on the map and a direction, the system will create and play a corresponding musical sequence, by selecting the tracks from the media library.

2.2.4 reacTable: the (infrared) light of demeTouch's physical setup

In addition to the idea itself of offering the user an original interaction with digital content displayed on a table's surface, demeTouch will reuse one concept of the reacTable project that concern the physical setup. The actual tabletop display will be improved by the use of an infrared lightning system that will prevent the camera from recording what is projected on the table's surface.

2.3 The application's idea: demelo

2.3.1 A new approach to music

The project demelo started in October 2005. Its concept came from three statements: first of all, although personal audio libraries are bigger and bigger, they usually suffer from a lack of classification. Finding a specific song among the huge amount of collected files asks the user a lot of patience which often results in her listening to the selection of most played songs. Secondly, editorial tags embedded into most music files (MP3, WMA, OGG...) are often incomplete, inaccurate or simply unused, and updating each and every file of its own library is a long and boring labor. Finally, a closer look at music stores' shelves reveals that famous artists are usually put forward while others are more difficult to access. Moreover, it is easy to associate an artist with a music genre (pop, ska, folk, afrobeat...) when this artist can embrace several different styles. Even an album from the same artist usually presents pieces of music differing in genre. In addition genres can sometimes be inaccurate due to their number, the difficulty to distinguish some of them and their subjectivity (one's interpretation will classify the ban Led Zeppelin in hard rock when others would say folk).

As described in [JFG06] (unpublished), the basic idea behind demelo is to get rid of these difficulties by offering the user a new approach to her library: demelo associates pieces of music with colors. The color is given by three musical descriptors extracted from the songs (style, dynamic and energy) which once converted to the color space and combined together give the music's color. This simple association helps in at least three ways:

- a huge unclassified library can be easily sorted by color: pieces near in color are acoustically similar
- artists are classified upon their musical style instead of their fame
- music search engines can provide at a same level every pieces of music corresponding to the user's request

2.3.2 The project into details

The following description of demelo introduces the reader to some ideas that will be reused later when designing demeTouch. However some more details concerning demelo's application are provided in appendix B.

Music player

Figure 2.6: Interface of demelo's music player (May 2006)

As depicted in Figure 2.6 this interface contains all basic commands needed when playing music: play/pause, navigation through the song and the playlist. It also offers some extra features: a frequencies equalizer and a graphical representation of the current track's dynamic.

Library manager

The library manager collects and displays all pieces of music known by the system. It is possible to add and remove entries from the collection. Each file is analyzed when added: fingerprint extraction checks if the song is already present in the library and if not, the three musical descriptors (style, dynamics and energy) are extracted to assign a color to the new file. The bottom part of the screen is reserved to display information about the selected file: editorial tags, CD cover (if known) and musical descriptors.

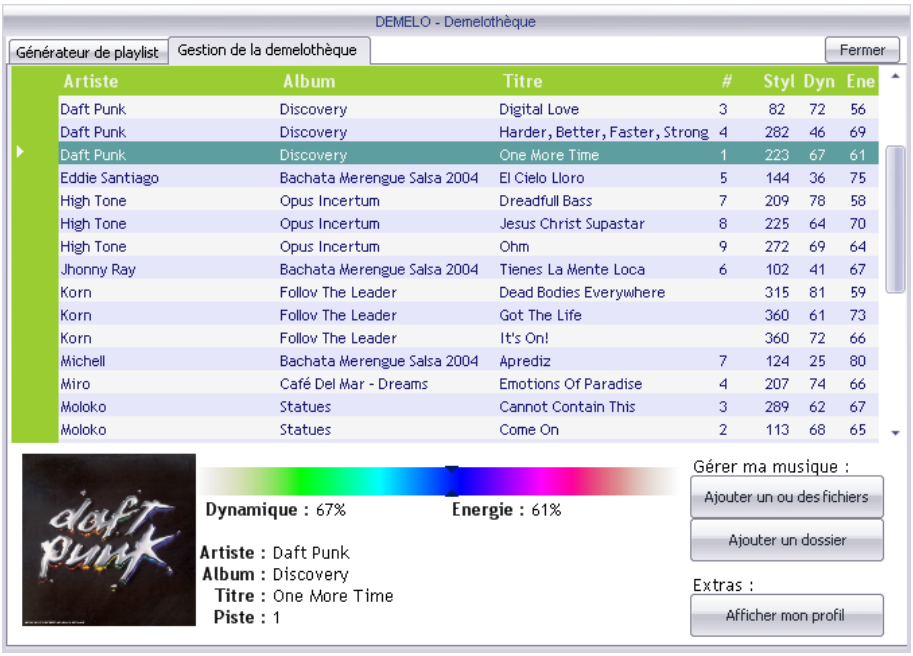


Figure 2.7: Interface of demelo's library manager (May 2006)

Playlist editor

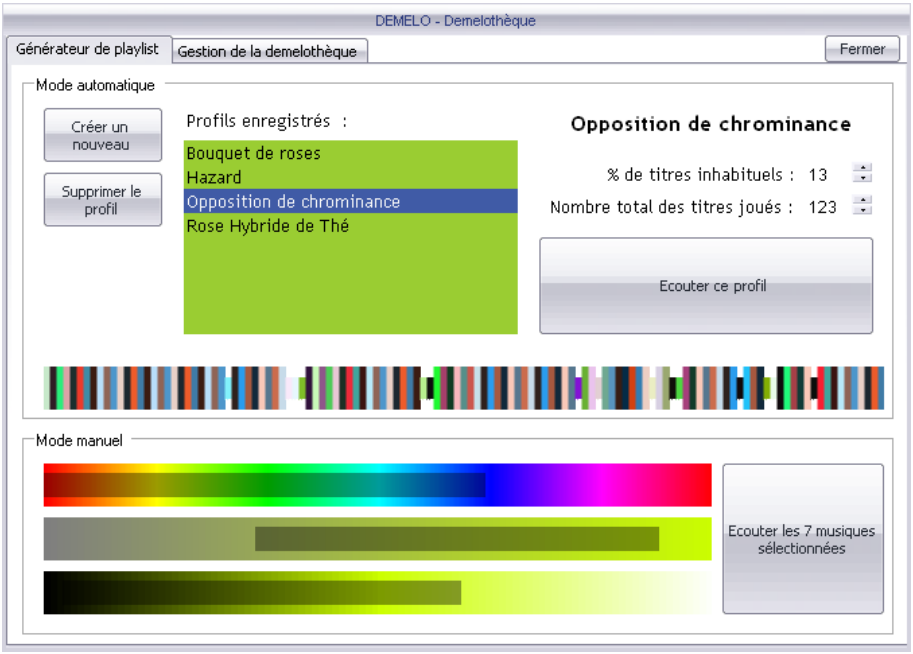


Figure 2.8: Interface of demelo's playlist editor (May 2006)

This feature allows the user to manage the sequence of colors that will be used to create a playlist. Profiles can be used to load and save sequences. To select a range of colors, ones should use the bottom part of the window. See appendix B for the details concerning the color selection.

Hardware

Figure 2.9: demelo's hardware first design (May 2006)

The hardware connected to the computer running demelo's software has two functions: playing the music and creating an atmosphere in accordance to the music played by being lit up according to the color of the song. The first design consisted on a combination of separate modules: a plexiglass tube in charge of the lightning and displaying information about the music played on a LCD screen, and two speakers in charge of playing the music. These modules were connected to the computer using a unique USB cable, although the idea was to replaced it by wireless USB connection to offer more freedom on the modules' place in a room. A second design is currently being developed by another team of student from the Ecole Centrale d'Electronique. Their idea is to create a lightning table to create the atmosphere in accordance to the music diffused by a separate sound system.

2.3.3 demelo: demeTouch's predecessor

The demelo project will be partially reused for demeTouch's application layer: the color extraction algorithm will be reused directly while the application will be redesigned to suit the new constraints and offer new possibilities to the final user.

CHAPTER 3

Analysis

Music is usually an important aspect of someone's personal environment. This chapter will first study the use of demeTouch in a real environment, as part of a home sound system. From simple use cases the basic requirements that the system will have to respect will be deducted. Then, a comparison will be done between a system as it could be designed in ideal conditions of time, workforce and technology and the system developed under the master's thesis constraints. Finally, the technologies used for designing the system will be presented.

3.1 Essential use cases

The use cases assume the two following statements:

- The system is ready to use: the camera and projector are plugged in and connected to the running computer where the application is installed.
- The user knows the system: she knows its purpose and what to expect from it. She knows how to navigate through the application and has a good knowledge about the music-color association.

3.1.1 Playing music according to a specific mood

The *Introduction* chapter stated that the system allows the user to select music taking into consideration her mood only, avoiding external distractions such as habits, time to look for specific songs, imperfect knowledge of her record library... In this use case the user is considered at home, in a mood for listening to some quiet music while reading a book.

Starting the application

The user stands in front of her computer. When she launches the application, the graphical interface is displayed directly on the table's surface. The user moves in front of the table, and sees the classical interface: the menu at the bottom of the table's surface with the *Player mode* selected, the main panel above the menu showing a simple audio player.

Creating the playlist

As she wants to listen to a specific color of music, she selects the *Playlist editor mode* in the menu. The main panel switches to a color-chooser interface. From the color-space representation, she selects a value in the greens area (associated music genre: jazz), with low dynamics (monotonous music) and low energy (quiet music). The system displays some information concerning her choice: the number of matching entries in the library, the sequence's duration and some examples of artist and songs. She decides that the sequence is not long enough, so she modifies her selection, by adding more dynamics: the selection now comprises pieces of music with richer structure, which suits her actual taste. She confirms her choice and leaves the table, grabs her book, sits down and starts reading.

Playing the selecting sequence

After the user's confirmation, the system switches to the *Player mode*, starting to play the titles from the media library that correspond to the user choice. At the same time, the table's surface is lit up by the color corresponding to the currently played song, emphasizing the room's atmosphere.

3.1.2 Creating playlist for a longer period

This use case places demeTouch in another classical environment: a host (who will be refered as the primary user) invites some guests (who will be refered as the secondary user) for a dinner. The system will be used to define a long playlist that is supposed to last for the whole evening: lounge music at the begining (while guests are arriving) then quiet music (during dinner) and finally more punchy music including popular music to eventually dance on.

Creating the sequence

Waiting for her guests, the primary user stands in front of the table's surface. The application displays the *Playlist editor mode*. She successively selects colors that correspond to lounge low-beat music, quiet jazz, slow electro, punchy electro, pop/rock and electro/rock music. Then she adjusts the values of dynamic and energy to start with calm and quiet music to end with more punchy and energetic atmosphere. When the first guests arrive, she confirms her choice and the system starts playing the selection, the application switching to the *Player mode*.

Switching to manual mode

After dinner, the primary user wants to organize a "blind test" game: she will play some famous music from the 70s and the audience will have to guess the artist and the title of the played music. To do so, she switches the interface from *Player mode* to *Media Library mode*, selects the songs she wants to play and clicks on the *Play sequence* button, which launches the playlist manually created.

3.2 Basic requirements

The previous use cases illustrate two typical use of demeTouch in an everyday environment. They provide useful information about the design of the system. The basic requirements inferred from these use cases concern the three main subsystems: the physical setup, the video processing engine and the application. If some of these requirements have already been introduced in previous sections, they are all of equal importance regarding the project's issues.

3.2.1 The table: a built-in piece of furniture

The two previous mini-stories take place in a personal living room, where people spend a lot of time, and where design plays an important role in term of furniture integration. demeTouch's table should offer the possibility for several persons to stand at the same time around the table. In addition, it should be built in a multi-usage way, so that people can use it as a regular table.

3.2.2 The video processing engine: a robust finger detector

Two main characteristics are expected from a finger-controlled system: speed and precision. First, speed is necessary to assure a latency-free response from the system to the user command. Taking the previous use cases as examples, the system should play a song as soon as the user presses the *Play button*, without leaving her finger for three seconds on the tables's surface. Secondly, precision helps avoiding the unpleasant situation of the system misinterpreting the position of the user's finger, resulting in a wrong action. For example, the volume should not be adjusted when the user clicks on the track bar.

3.2.3 The music application: an intuitive interaction with the system

From the two previous use cases it seems that the application requires to rely on strong visual conventions not to loose the user in an interface difficult to understand, when the way to interact with it (using fingers on a table's surface) is already uncommon. For example, buttons should look like classical buttons with explicit labels, so that even a new user should feel familiar with the application. In addition, a *Help section* should be accessible to explain how to navigate thought the interface and to detail every component's action.

3.3 Ideal system vs. real system

3.3.1 A ready to sell version of demeTouch

In ideal conditions of time and workforce, demeTouch could achieve a level of development that would eventually make it ready for entering the market of innovative solution for human-music interaction.

First of all, the table would be built in a less massive way to make it more integrable in a home environment. The material used to build the table would have noise attenuation properties, to remove the sound caused by the projector's ventilation. Secondly, the physical elements (the projector, the camera, the infrared lightning system) would be surely attached to the table's structure so that the calibration issue would be simplified and the table could be easily moved. In addition, the setup would include a temperature regulator to decrease the heat caused by both the infrared lightning system and the projector. Finally, the active surface would be made of resistant material: for example a shatterproof washable plate-glass.

The video processing engine would be improved to offer a high recognition rate on the whole active surface. The finger detector would locate fingers touching the table's surface with high accuracy. It would adapt itself to the conditions of luminosity and adjust the processes' parameters in consequence. In addition, it would allow the use of tangibles in addition to fingers, taking advantage of the video processing's flexibility.

The application would offer the user a colorful experience of music, by making use of the whole table's surface in a beautifully designed fashion. Navigating through it would be natural and elegant. For example it would let the user choose what and where should be displayed. It would include nice extra features like a visual render of the selected playlist, a painting tool that would allow the user to "draw " a sequence matching her mood... The programmer's visual style (rigid and rather bad-looking) would be replaced by a light artistic graphical user interface, that would make the user enjoy using demeTouch.

3.3.2 Some room left for improvements

Unfortunately, the constraints imposed by the master's thesis project in terms of time imply to set priorities and to put aside some ideas. The aim being to end up with a working solution, even if it is a simplified one.

The table itself will be reused as it was at the beginning of the project. Adding the camera and the infrared lightning system will be the only modifications done to the physical setup. It means that calibration will be a real challenge, as a small change in the elements' location would significantly modify the finger detection accuracy.

The video processing engine will deal with fingers only. The parameters involved into the finger detection will be manually adjustable, so that the user will have a total control on the video processing engine, to the expense of the "automation" aspect. In addition, a specific part of the table's surface only will be active, meaning that the user will not be able to interact with everything that is displayed.

The application will be a demo application aiming to illustrate the possibilities offered by such a setup. The accent will be put more on the functionality than on the visual aspect. The basic functions will be implemented, leaving room for the implementation of extra features.

3.4 Technologies review

This section introduces the technologies that will be later used to design the system: tabletop displays are presented, explaining the capabilities of such a technology illustrated by recent examples. Then some basics in image processing are provided to the reader. Finally the music-color association, the main idea behind the application, is introduced from an historical point of view.

3.4.1 Tabletop displays and multi-touch screens

Tabletop displays

Tabletop displays are used to display information to users on a table's top surface. Although some systems are dedicated to a single user, such systems are mostly used for collaborative work, especially during meetings: every participants located around the table can access the information displayed on the table. Evolved systems allow users to interact with the displayed content, improving considerably the efficiency of a brainstorming meeting: while documents are exchanged and discussed, users can edit, copy, save them in live. Figure 3.1 presents an example of interactive tabletop displays clearly dedicated to collaborative work. The reader can refer to [CIS+06] for further information concerning this system.

Figure 3.1: An example of tabletop display used for collaborative work
Photo © Calgary SUN

Of course, such devices can be used for other purposes than collaborative work: learning, art creation, gaming. The state of the art example of a tabletop display dedicated to entertainment is provided by Microsoft Surface[texttrademark][1]. This 30-inch device offers to several users surrounding the table the possibility of manipulating pictures, videos, sending emails, consulting itineraries...

Multi-touch screens

Wide multi-touch screens can also be used for document sharing and manipulation. An example of such a device is provided by Perceptive Pixel[texttrademark]. The impressive demonstration video[2] illustrates the use of the device for sharing and manipulating documents, data mining, photo editing... The first advantage that appears from this demo is the intuitiveness of the system: users open, rotate, copy, zoom on documents with simple finger movements in a very smooth way. Another advantage comes from the interface itself: it is fast and attractive. The system must relies on a robust finger detection and position interpretation to allow the developer to create applications offering nice design.

Why a tabletop display?

The examples provided in this section are an introduction to the concept of displaying and manipulating information on an exotic surface. This idea is part of demeTouch, which uses the table described in section 2.1. The Perceptive Pixel[texttrademark] and Microsoft Surface[texttrademark] projects also demonstrate the importance of both the system's intuitiveness and the interface's design in term of attractiveness. I will take these remarks into consideration when designing demeTouch's interface. The use of a back-projected tabletop display instead of a multi touch screen has been guided by two factors: flexibility and price. Indeed, tabletop displays based on a projector/camera system can be designed for applications that use any type of tangible as an input (like reacTable) and not only fingers. Which means that demeTouch can be improved later by implementing an object-based interaction. In addition, wide multi touch screens are still extremely expensive, whereas demeTouch relies on a rather cheap physical setup.

3.4.2 Object recognition in video processing

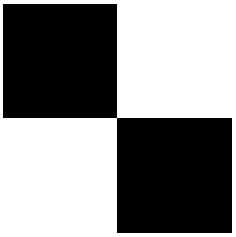
Recognizing defined objects in a scene requires to address issues inherent to both the objects to be recognized and the scene they should be extracted from. *Does the object*

[1] Further information can be consulted at `http://www.microsoft.com/surface/`
[2] Further information can be consulted at `http://www.perceptivepixel.com/`

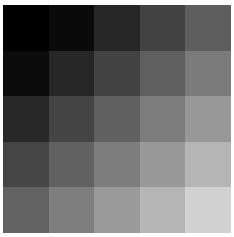
have a simple shape (geometrically speaking) and can be represented in an easy way, or does it present randomly distributed irregularities? Is the scene highly contrasted or is it very noisy? Should the treatment be done in real-time? These are some questions that should be answered before starting solving an object recognition problem. Indeed, each problem is unique although most of them could be solved by the use of several different techniques. For that reason a careful study of the problem's properties should help finding the most efficient method. Next paragraphs discuss some of these techniques, illustrated by examples. First, basic notions commonly used in image processing are presented, to simplify the reading of future parts of the report.

Basic notions and tools used in image processing

Images representation The most common way to represent an image is to consider it as a 2-dimensional array of numbers. Each of them standing for a pixel value, as depicted in Figure 3.2. Most frequently images are considered in a gray scale format, with pixels values from 0 (black) to 255 (white). Indeed, when a treatment is applied to the image, a predefined number of pixels (usually all of them) are scanned in a loop on their number. Thus, representing them as an array of gray scale (8 bits) values considerably simplify the manipulation of the image. Beside, from a computer perspective, processing 8 bits images requires less memory and is less time consuming than processing "real color" (24 bits) images. Figure 3.3 illustrates these considerations.



$$\begin{pmatrix} 0 & 255 \\ 255 & 0 \end{pmatrix}$$



$$\begin{pmatrix} 0 & 10 & 38 & 66 & 94 \\ 11 & 39 & 67 & 95 & 123 \\ 40 & 68 & 96 & 124 & 152 \\ 69 & 97 & 125 & 153 & 181 \\ 98 & 126 & 154 & 182 & 210 \end{pmatrix}$$

Figure 3.2: Images are represented as an array of pixel values. Here are two examples with their corresponding array representation.

Image dimension	Array size (kb)	
	Color image	Gray scale image
$95 \times 113 = 10735$ pixels	257.64	85.88
$640 \times 480 = 307200$ pixels	7372.8	2457.6

Figure 3.3: Considering an image as a gray scale reduces the size of the array by a factor 3

Usually, computers consider this array of pixels in a one-dimensional array. Then, to map a pixel's coordinates from two dimensions to one, formula 3.1 should be used.

$$x\_coord_{1dim} = x\_coord_{2dim} + imagewidth * y\_coord_{2dim} \tag{3.1}$$

Processing an image Processing an image is the action that creates an output image B from an input image A. To do so, each pixel in A is successively read and an operation is applied to it that will result in the creation of a new pixel in B. Figure 3.4 illustrates the operation of processing an image.

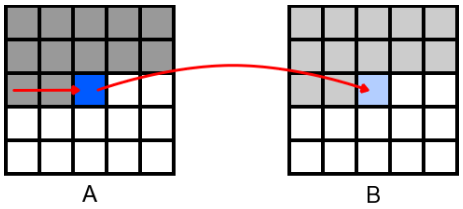


Figure 3.4: Illustration of pixel-by-pixel image reading

Types of operations Three types of operation can be applied to the original image, depending on the operation's goal:

- **Point operations:** each pixel in B is created from the pixel in A at the same location.
- **Local operations:** each pixel in B is created from the pixel in A at the same location and its neighborhood.
- **Global operations:** each pixel in B is created from the pixel in A at the same location and the whole image.

Table 3.1 indicates the complexity of each technique considering a $N \times N$ image and a $n \times n$ neighborhood, illustrating each type with an example. From this remark it can be deduced that for a real-time application, point operations are considerably more efficient than global ones.

Types of operation	Complexity per pixel	Example of operation
Point	$O(k)$ (constant)	Binarization using simple threshold
Local	$O(n^2)$	Edge detection using convolution
Global	$O(N^2)$	Image reconstruction using filter point spread function

Table 3.1: Complexity of the three types of operation

In addition to these three types of operation, the authors of [CP95] distinguish two operation levels:

- **low-level operations** are applied in a pixel-by-pixel manner, each pixel and eventually its neighborhood being processed without considering the meaning of the operation or the result of this operation. Convolution is an example of such an operation, as it deals only with the numerical nature of the processed data.
- on the contrary **high-level operations** deal with the symbolic nature of the data. Images are computed regarding the goal of the process: for example in an aerial view of a landscape the extracted shapes can be annotated depending on the nature of the shape: building, road... Then when the connections between these shapes are processed, assumptions like "a road cannot go through a building" can be adopted.

Types of neighborhood When using an operation that require to consider the neighborhood of a pixel, two kind of sampling can be differentiated: rectangular and hexagonal. Inside these two categories the number of considered pixels varies depending on the desired size of the neighborhood. Figure 3.5 illustrates three common examples of neighborhood.

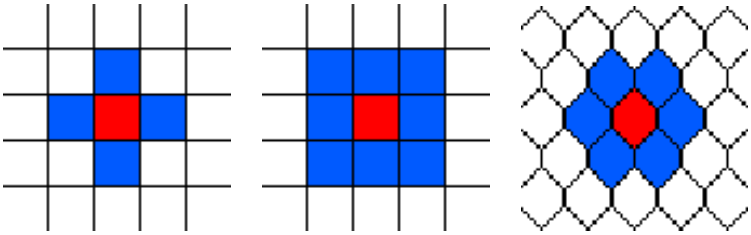


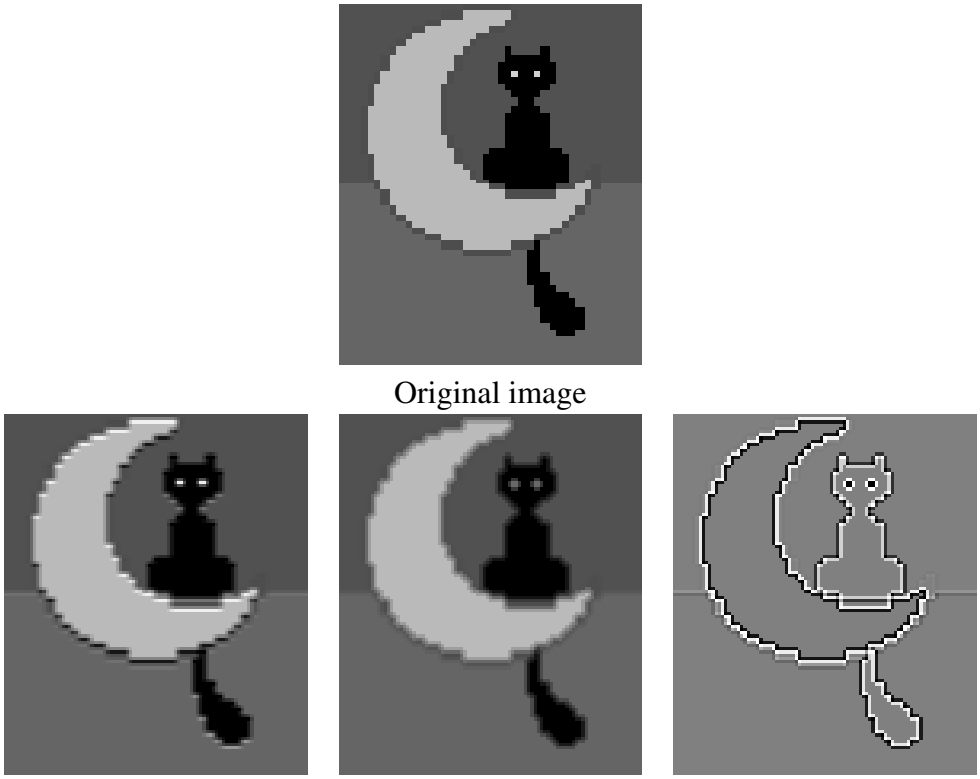
Figure 3.5: Three common neighborhoods: 4-connected rectangular (*left*), 8-connected rectangular (*center*) and 6-connected hexagonal (*right*)

Convolution This mathematical operator is essential in image processing. It consists on a small window propagated along the image's array of pixel values. For each pixel read, it processes the weighted sum of the pixels within the window. The window is called the convolution kernel and represents the weights applied to the pixel values. Mathematically, this operator is represented as in equation 3.2 where A is the input image, k the kernel and B the output image.

$$B = A * k \tag{3.2}$$

Such an operator can be used for different purposes, such as sharpening, smoothing, detecting edges, removing noise... Figure 3.6 presents the result of three convolution operations applied to the same image, indicating the kernel used.

Fourier Transform This transformation allows to switch from a spatial-based to a frequency-based representation of the image's signal. Digital image processing uses the Discrete Fourier Transform (DFT), which is a sampled Fourier Transform. Instead of the full range of frequencies that form an image, it takes into consideration only a set of samples, large enough to fully describe the image's spatial domain. Fourier Transform is commonly used for image analysis, image filtering, image reconstruction



Original image

Sharpening filter
$$\begin{pmatrix} -1 & -2 & -1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$$

Smoothing filter
$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

Edge detection filter
$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

Figure 3.6: Three examples of convolution applied to the same image. For each example, the corresponding kernel is indicated

or image compression. This operation is particularly useful in noise removal processes: considering a noisy image and knowing the noise's frequencies, it can be easily removed from the transformed image, resulting in a denoised image.

Basically, the two-dimensional DFT of a square $N \times N$ image is given by equation 3.3, where $f(m,n)$ is the spatial domain image and the exponential term is the basis function corresponding to each point $F(m',n')$ in the Fourier space. This equation can be interpreted as follow: each point $F(k,l)$ is obtained by summing the result of the multiplication of the spatial image by the corresponding base function. In practice, to compute the DFT of an image, it is more efficient to use a Fast Fourier Transform algorithm, which reduces the complexity of the operation from $O(N^2)$ to $O(Nlog_2(N))$.

Some properties of the Fourier Transform are very useful in image processing: for example a multiplication in the Fourier space corresponds to a convolution in the spatial space, which makes edge detection easier to process. Figure 3.7 illustrates this concept, by showing an image with its Fourier Transform, and the reconstructed image after applying an ideal low-cut filter to the transformation. In fact, edges are characterized by a rapid change of pixel values, which correspond to high frequencies in the Fourier space. By removing the low frequencies in the Fourier Transform representation of the image, only zones with rapid change in pixel values are preserved.

$$F(m',n') = \frac{1}{N^2} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} f(m,n) e^{i2\pi\left(\frac{mm'}{N} + \frac{nn'}{N}\right)} \tag{3.3}$$

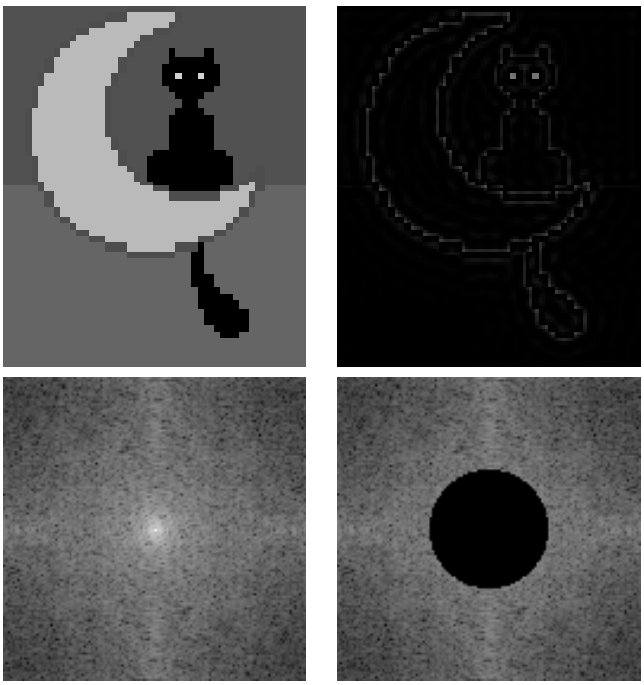


Figure 3.7: An example of edge detection using Fourier transform

Histograms Another manner to represent an image is to build its pixel values' histogram. The pixels' distribution is deduced from such a representation at one glance, by looking at the shape of the histogram: peaks stand for the most frequent pixel values in the image. Figure 3.8 illustrates the use of such an histogram for thresholding an image.

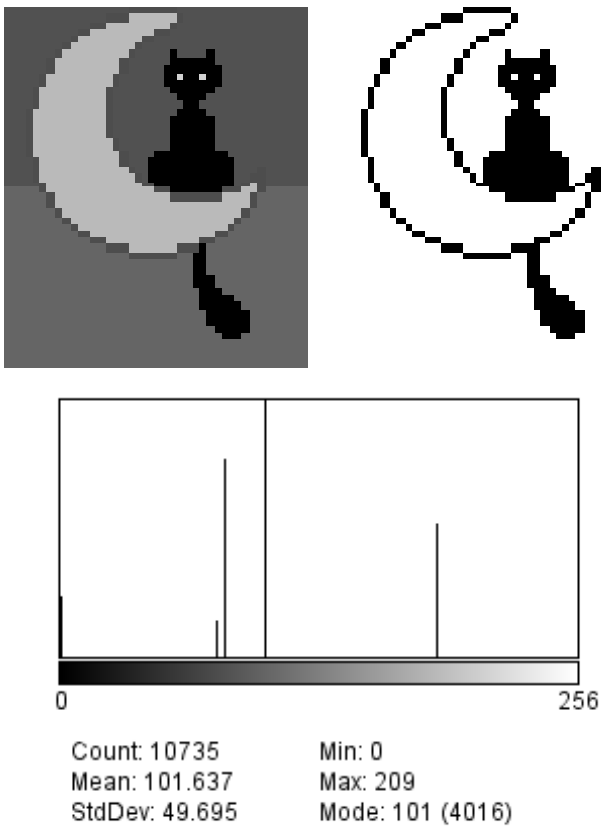


Figure 3.8: An example of pixel values' histogram used for thresholding

Video processing in demeTouch

Processing the images retrieved from the camera will be one of demeTouch's major concern. The video processing engine will have to be robust enough to clearly identify fingers from the noisy scene, and fast to do the process in real-time and in background of the front-end application, which will be running simultaneously.

3.4.3 The color-music association

An old concept

Associating music with colors have been discussed by scientists since the XVIIth century. Newton was one of the first to compare the musical scale to the color scale: the three primary colors (blue, red and yellow) plus the three secondary colors (green, or-

ange and violet) plus indigo can be mapped to the seven notes that form the musical scale (see Figure 3.9).

Figure 3.9: Seven-scale music-color association by Newton

Later, Louis-Bertrand Castel completed this approach by associating the twelve-note chromatic scale with twelve colors as depicted in table 3.10. He went even further by inventing the *Ocular Harpsichord* described in details in [Fra91]. The basic idea of the instrument was to offer to deaf artists the possibility of *feeling* the music by representing it using colors.

Figure 3.10: Twelve-scale music-color association by Castel

Since that time, scientists improved Castel's studies to optimize the correlation between what people hear and what they see. More recently, studies in the chromotherapy field tended to demonstrate the impact of colors on people's mood. The reader can refer to [AR05] for an introduction to chromotherapy, its principles and applications. Although this interpretation is rather subjective and can vary from a culture to another, Morton Walker demonstrated in [Wal91] that colors affect people's feelings. Table 3.2 summarizes his interpretation, that applies for most western countries.

Color	Associated feeling
Red	Passion
Orange	Attention
Yellow	Stress
Pink	Weakness
Blue	Strengh
Green	Calm
Brown	Security
Black	Depression
Gray	Neutral

Table 3.2: Morton Walker's color-mood association

Musicovery: a recent application of the music-color association

Musicovery[3] is an excellent example of a color based playlist creator. If offers the user the possibility to create a sequence of songs that correspond either to common musical genres (jazz, rock, electro...) each of them being associated with a color, to years (from the early fifties to today) or to a musical mood (dark, energetic...). As depicted in 3.11, once the playlist is created, a colored "tree" appears representing the sequence of songs, each of them displayed in his color and linked to the previous and next ones.

Associating music with colors: demeTouch's core idea

One of the demeTouch basics is to offer the user a color-based representation of her audio library. Combining the music-color association with the color-mood association gives a new potential to demeTouch: harmonizing the played music (selected upon the user's mood) with the visual render of the application. By lightning the table's surface in the color corresponding to the music played, the user's mood will be emphasized both in the auditory and visual environment.

[3] `www.musicovery.com`

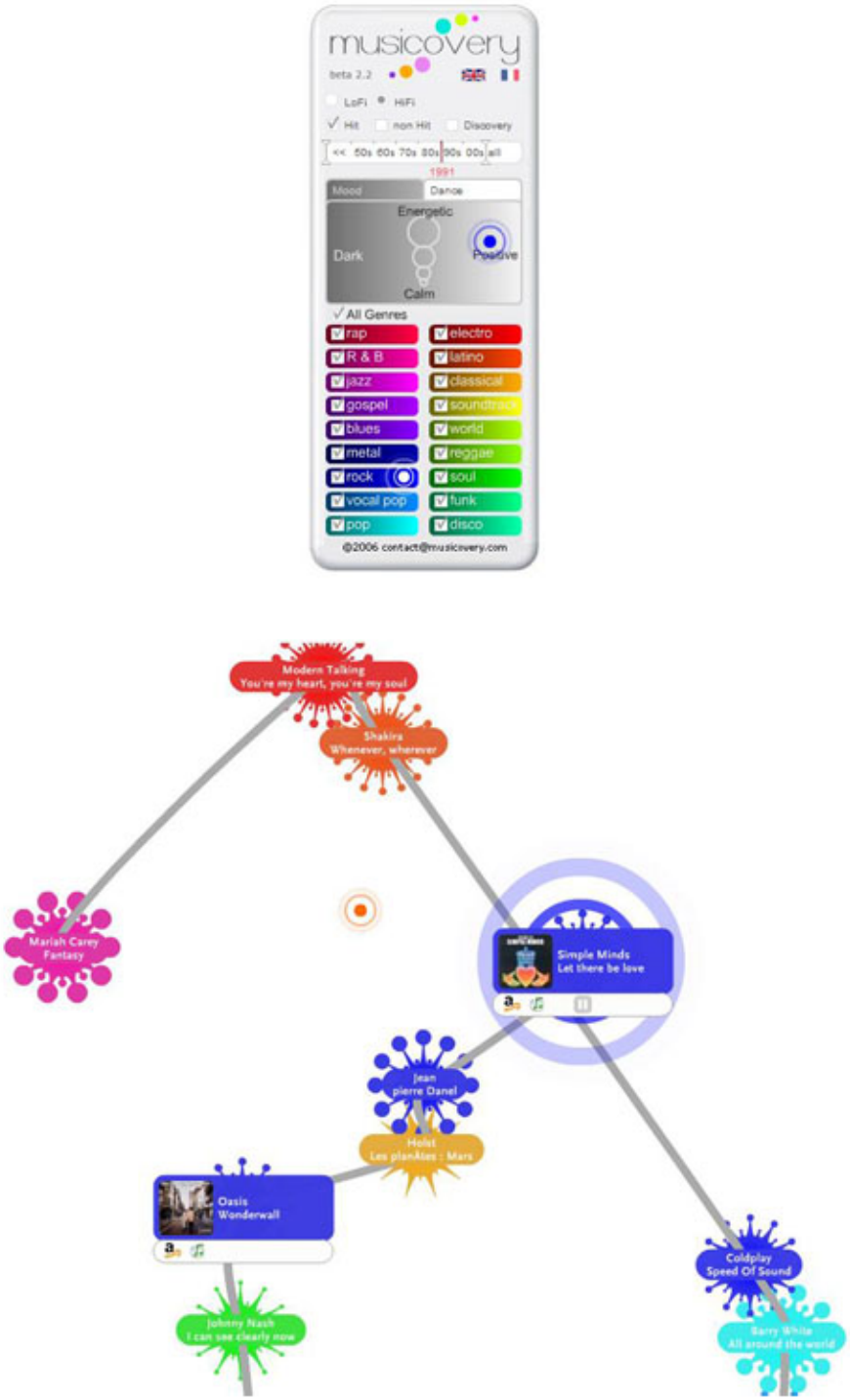


Figure 3.11: Musicovery's interface: the "remote control" is used to create the playlist represented in a tree-like manner

3.5 Image processing tools

3.5.1 Building methods for detecting fingers

As introduced in chapter 2, several techniques can be used to extract fingers from a scene. The basic requirements previously stated in this chapter impose the use of robust but fast methods: point and local low-level operations with a small neighborhood (8-connected at the maximum) are preferred to global or local high-level operations that consider a large neighborhood and require to understand the meaning of the detected objects. In addition, operations should be combined when possible to avoid reading the array of pixels several times if it is not absolutely necessary. The following sections present separately the six operations that have been developed in collaboration with Frédéric Boulet. These basic methods will be later adapted, eventually associated together and successively applied to the images provided by the camera, when it will come to design the video processing engine.

Subtraction

This operation processes a pixel-by-pixel subtraction between the current image A and a reference image R resulting in a new image I. Equation 3.4 shows the process applied to the pixel i located at the coordinates (x,y). The absolute value is used to avoid the problem of negative pixel values, that does not have a relevant computational meaning in the present situation.

$$i_{(x,y)} = \mid A_{(x,y)} - R_{(x,y)} \mid \tag{3.4}$$

For the purpose of finger detection, the reference image should correspond to the scene without fingers laying on the table. In that case, the ideal resulting image would have most pixels set to 0 (as most of the image does not change); only the zones where fingers appear being set to a non-zero value. In reality, due to the noise introduced by the camera (characterized by a sparkling effect), the current image is never perfectly equal to the reference one, even in zones without fingers. To handle this problem, a threshold ε_S is introduced so that if the subtraction result is less than ε_S, the two images are assumed equal. ε_S is ranged from 1-255. Finally, the subtraction operation can be resumed by equation 3.5. This equation is illustrated in Figure 3.12, showing that when the difference between the reference and the current image is below a threshold, the pixel's value is set to 0, and to the calculated difference otherwise.

$$i_{(x,y)} = \begin{cases} \mid A_{(x,y)} - R_{(x,y)} \mid & \text{if } \mid A_{(x,y)} - R_{(x,y)} \mid > \varepsilon_S \\ 0 & \text{otherwise} \end{cases} \tag{3.5}$$

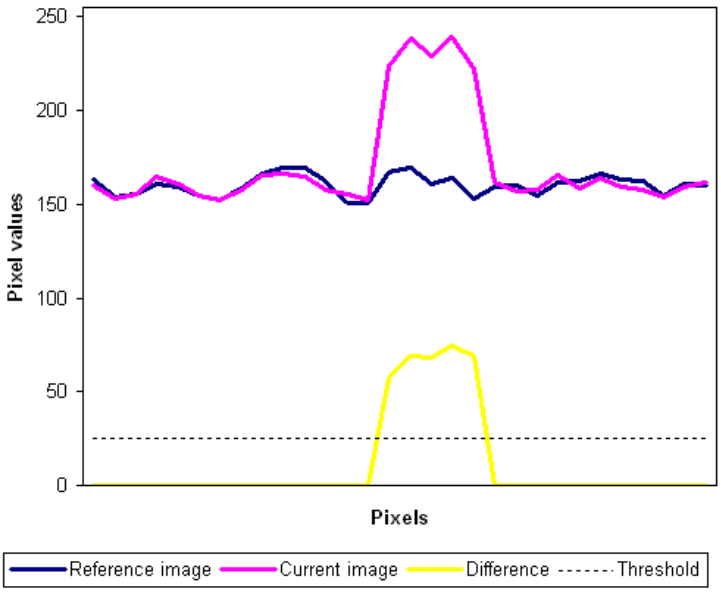


Figure 3.12: Subtraction illustration: the result pixel values are either 0 or the difference between the reference and the current image

Binarization

Binarizing an image consists in setting its pixel values either to 0 (black) or 255 (white). Basically a threshold ε_{Bi} is defined and pixels with a value less than ε_{Bi} are set to 0 while pixels with a greater value are set to 255, as shown in equation 3.6.

$$i_{(x,y)} = \begin{cases} 255 & \text{if } A_{(x,y)} > \varepsilon_{Bi} \\ 0 & \text{otherwise} \end{cases} \tag{3.6}$$

In the present configuration, the two sources of infrared light are reflected by the table's surface, and these specular reflections are also filmed by the camera. Thus they appear on the video as two extremely bright spots (in these areas, pixel values are very close to 255). The binarization process finally uses two thresholds ε_{Bi_d} (threshold down)

and ε_{Bi_u} (threshold up), so that only pixels with a value comprised between these thresholds are set to 255, as formulated in equation 3.7.

$$i_{(x,y)} = \begin{cases} 255 & \text{if } \varepsilon_{Bi_d} < A_{(x,y)} < \varepsilon_{Bi_u} \\ 0 & \text{otherwise} \end{cases} \tag{3.7}$$

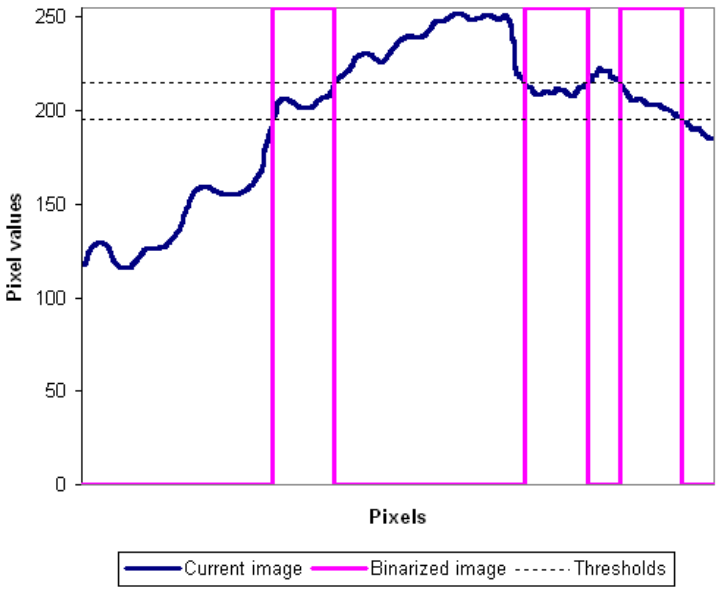


Figure 3.13: Binarization illustration: pixel values are set to 255 only if they are comprised between the two thresholds

As depicted in Figure 3.13, the limit of binarization is reached when extremely bright areas are present in the image to be processed and are not fingers. This noise can only be partly removed, otherwise the fingers' brightest part would also be discarded, resulting in an imprecise recognition.

Brightness enhancement

Basically this algorithm is used to help the binarization process. As it uses fixed thresholds for the whole image, binarization depends on the infrared beams repartition on the table's surface. The more infrared light, the more reflection from the fingers, and as a consequence the more visible. Unfortunately, infrared beams are not equally spread

on the table's surface, making the fingers less reflective in some areas. Increasing the image's brightness consists in adjusting pixel values so that darker areas that show fingers are not discarded. In fact, without this process, brightest pixels in darker zones are usually not bright enough and are discarded by the binarization. The operation works as follows: the current image is divided into windows, inside of which pixel values are normalized. To do so, every pixel value is multiplied by the factor that transforms the highest pixel's value of the zone into the value 255: $\frac{255}{maxValue}$. Equation 3.8 gives the mathematical definition of this operation applied to an image divided into K windows. The size of windows is set so that the specular reflexions fill a unique window, which is about 40×30 pixels big in the present configuration.

$$i_{(x,y)} = \phi^k \cdot i_{(x,y)}, \text{ where } \phi^k = \frac{255}{\max i^k_{(x,y)}} \tag{3.8}$$

The drawback of this method is that dark zones are normalized even if they do not show fingers, which can lead to considering bright spots as fingers when they are just noisy areas that have been amplified. Nevertheless, in these dark areas the multiplication factor is high, as the brightest pixel has a low value. Thus a threshold ε_{Br} is introduced to take into consideration only windows where the factor is less than this threshold. Indeed, dark zones without fingers are considered as noise and discarded, while dark zones with fingers are kept. Illustrated by Figure 3.14, the final equation of this process is given in 3.9.

$$i_{(x,y)} = \begin{cases} \phi^k \cdot i_{(x,y)} & \text{if } \phi^k < \varepsilon_{Br} \\ 0 & \text{otherwise} \end{cases}, \text{ where } \phi^k = \frac{255}{\max i^k_{(x,y)}} \tag{3.9}$$

In Figure 3.14 only windows 2 and 5 potentially contain fingers. Considering the threshold down ε_d introduced in the binarization explanation, While window 5 is very bright and would have been correctly binarized, window 2 would have been considered as too dark and discarded. The brightness enhancement method should thus be applied before the binarization process, that would only take one threshold (ε_d) as a parameter. Modifications applied to the basic algorithms and the way they are combined are detailed in chapter 4.

Holes' removal

This process can be applied only after the image has been binarized. Like in the brightness enhancement method, windows are defined (smaller ones: usually 6×6 pixels big),

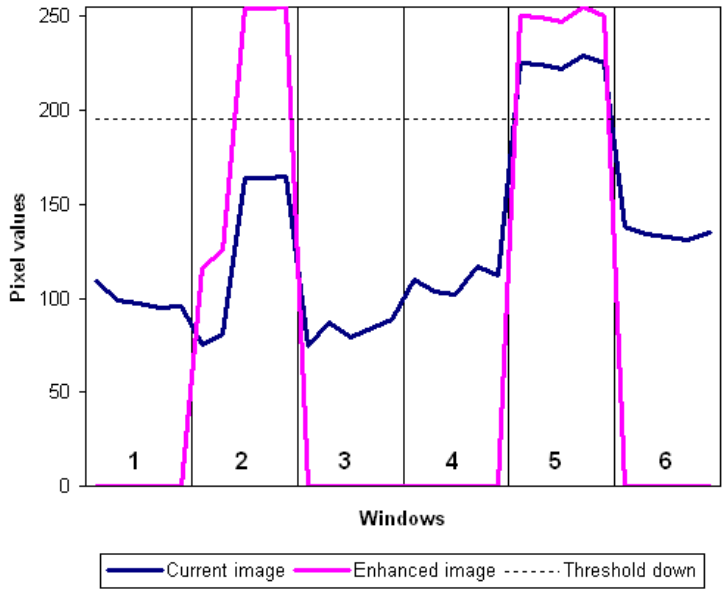


Figure 3.14: Brightness enhancement illustration: for each window, pixel values are normalized so that darker areas are not automatically discarded. The multiplication's factor is thresholded (ε) to prevent dark zones without fingers from being enhanced

inside which pixel values are counted (after binarization they are either 0 or 255). Then, all pixels in the windows are set to the most recurrent value, as illustrated by equation 3.10, where λ_δ is the number of pixels with value δ.

$$i_{(x,y)} = \delta\left(\max\left(\lambda_{\delta=0}, \lambda_{\delta=255}\right)\right) \tag{3.10}$$

This technique is used to make the finger representation a plain continuous area instead of a discontinuous set of spots. Figure 3.15 illustrates the process, where a 6×6 window (delimited by red squares) is used. The images represent two different spots that might be detected, a noisy one and a finger-like one.

Spots creation

After the previous methods have been applied, the original image is mostly composed of black pixels, only the areas showing potential fingers being white. The array of pixel

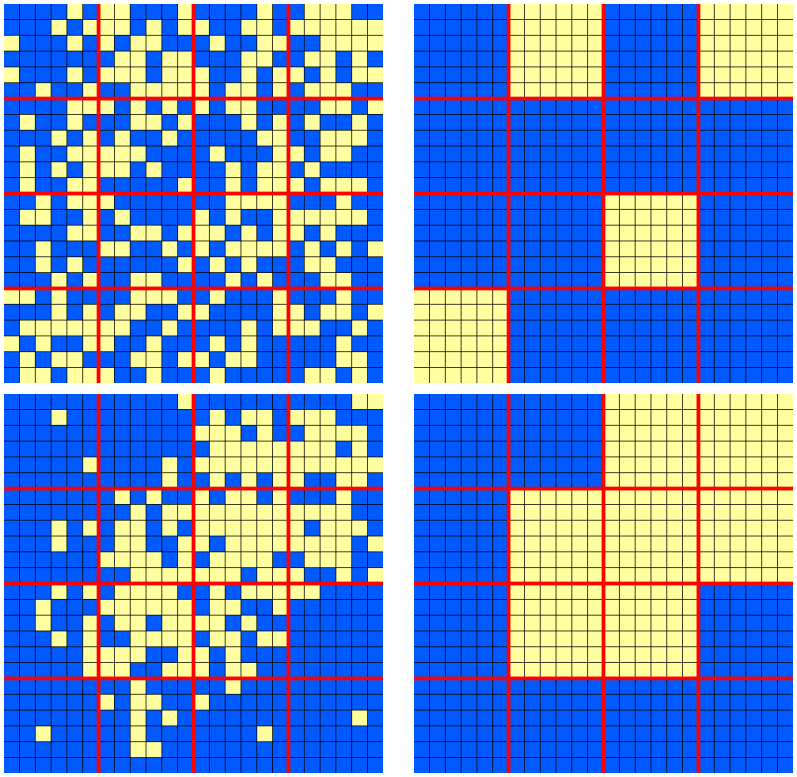


Figure 3.15: Holes' removal process: the extremely noisy spot in the *top left* image results in the discontinuous shape (*top right*), while the finger-like spot (*bottom left*) results in the well defined continuous area (*bottom right*)

values is scanned once more to extract the location of these potential fingers. During that operation, every white pixel launches the spot creation process, that consists in looking at the 8-connected surrounding pixels and eventually adding them to the spot if they are also white. Every time a pixel is added to the spot, it is marked to avoid taking it into consideration more than once. Indeed, as illustrated in Figure 3.16, without being marked pixels would be indefinitely added to the spot. In this example, pixel $(1,1)$ is first detected as the spot's initial pixel. Then surrounding pixels are scanned in the following order: top-left, top-middle, top-right, center-left, center-right, bottom-left, bottom-middle and bottom-right. The spot is then detected is the order shown in the right image in Figure 3.16.

At the end of the process, each spot s is defined as a collection of white consecutive pixels, as formulated in equation 3.11. The result of this process on the spots considered

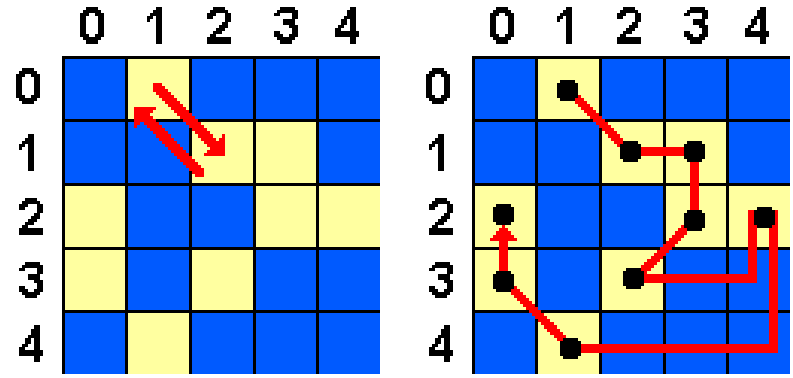


Figure 3.16: Spot creation's process: marking pixels while adding them to the spot prevent from entering in a infinite loop (*left image*) when the connected pixels are considered

in 3.15 (after dilatation) would be the following: the noisy spot results in the creation of four separate small spots, while the finger-like spot leads to the creation of a unique big spot.

$$s = i_{(x,y)} = 255 \tag{3.11}$$

Size discard

Once the spots are detected, some of them may have either an irregular shape (Figure 3.15) or a size that cannot correspond to a finger: extremely small or enormous spots have a very high probability to be due to punctual noise introduced by the camera or the projector. The set of spots S introduced during the spot creation's explanation is now regulated by two thresholds ε_d (threshold down) and ε_u (threshold up) controlling the spots' size, as shown in equation 3.12.

$$S = \sum_{k=0}^{K} \sum_{n=0}^{N_k} i_n = 255 \text{ if } \varepsilon_d < N_k < \varepsilon_u \tag{3.12}$$

Coming back to Figure 3.15, only the finger-like spot would be considered as a finger, the four spots in the other configuration would be considered too small.

3.5.2 Calibration issue

If the use of a mirror allows to increase the distance between the table's surface and the projector and camera, it also introduces a distortion angle increased by the setup's default. In an ideal configuration the camera and the projector would be perfectly aligned both horizontally and vertically so that their lens would be located in the very center of the configuration. They would film and project forming a right angle with the table's surface (see Figure 3.17).

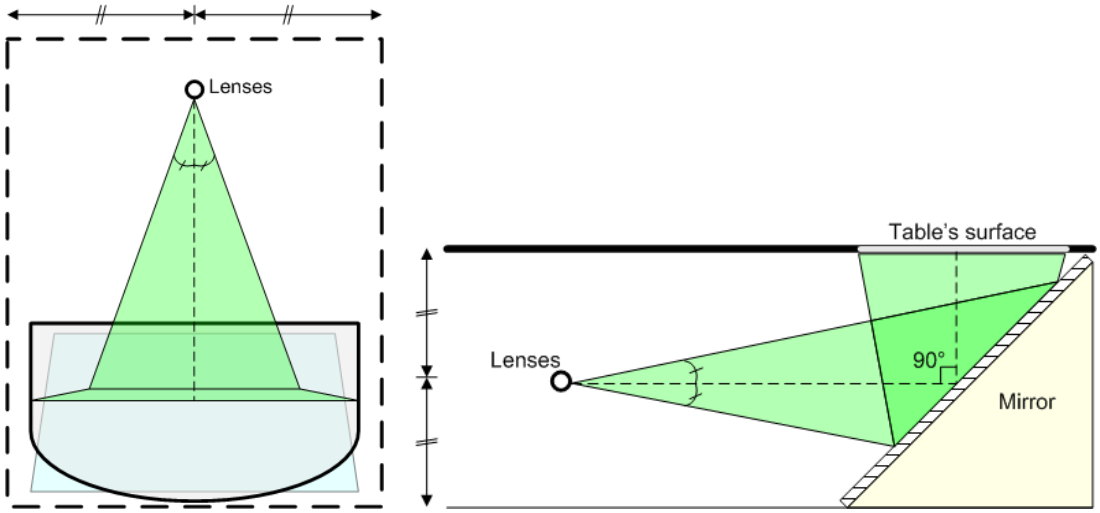


Figure 3.17: Ideal table setup: placing the camera and the projector in the ideal central position horizontally (*left image*) and vertically (*right image*) limits the distortion angle

Unfortunately, it is impossible in practice to set the projector and the camera at the exact same location. As demonstrated in chapter 4, both the camera and the projector present a vertical angle. It means that when a finger is detected by the camera, its recorded coordinates does not correspond to the projector's coordinates. If the projector's distortion can be corrected by adjusting its parameters, the distortion introduced by the camera has to be corrected manually. Figure 3.18 shows the impact of an imprecise positioning of the camera on the distortion angles (horizontal and vertical).

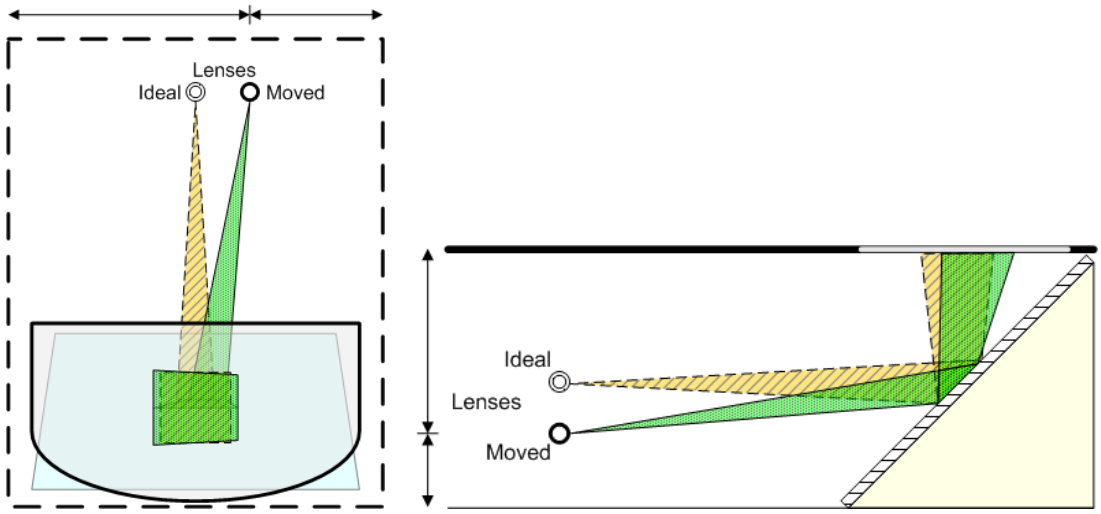


Figure 3.18: Distortion issue: moving the lenses horizontally or vertically from the ideal position introduces an angle that leads to imprecisions in finger recognition

General approach

If the physical setup cannot be perfect, it can at least be optimized: the camera and the projector are aligned horizontally, which decreases the horizontal distortion. In addition, the difference in vertical position between them is not extreme, limiting the vertical distortion. Finally, it is assumed that the zone filmed by the camera matches the projection zone. These statements lead to assume that it exists a point on the table's surface where a finger would be recognized at its real coordinates. Knowing this point $C_{(x_C,y_C)}$ the aim of the calibration process is to determine the function Φ that would allow to switch from the camera's coordinates system to the projector's one. ϕ is a function of the angle θ and the distance δ to C as shown in 3.13.

$$\Phi = f(\theta, \delta) \tag{3.13}$$

Simplified approach

In practice, the horizontal and vertical distortions are small enough to apply a simplified solution to the calibration issue. The procedure is simple: knowing their coordinates in the projector's system, a set of spots is displayed on the screen. Then their location in the camera's coordinates system are recorded. From these measurements a set of functions of the difference along by the horizontal and vertical lines is computed.

For each line, the mapping function is created by applying a curve fitting algorithm to the five spots forming the line. To do so a fourth order polynomial regression is computed, resulting in a function $f(x)$ like equation 3.14. The choice of the polynomial's order is important as it determines the future precision of the coordinates mapping operation. A fourth order regression is a very general solution and suits well a non optimized physical configuration.

$$f(x) = a + bx + cx^2 + dx^3 + ex^4 \tag{3.14}$$

CHAPTER 4

Design

This chapter first presents demeTouch from a general perspective, explaining the role of the different subsystems that form the overall system and how they interact together. Each subsystem is then detailed separately: the use of the tools defined in the *Analysis* chapter is explained, related challenges are exposed and solution are proposed.

4.1 System overview

Although it involves uncommon components such as a tabletop display, demeTouch's general design is rather simple: as depicted in Figure 4.1, three physical components interact together either in a unidirectional or bidirectional manner.

- The **tabletop display** is the interface between the system and the user: it gives the system information about the user interactions with the table (use of a finger to select, click...), and it provides the user a visual feedback of her action's impact on the system.
- The **local computer** is the central system's component: it processes the information provided by the camera, it allows the user to control the video processing engine, and it takes care of the visual and audio feedback.
- The **sound system** is in charge of playing the audio environment it receives from the local computer.

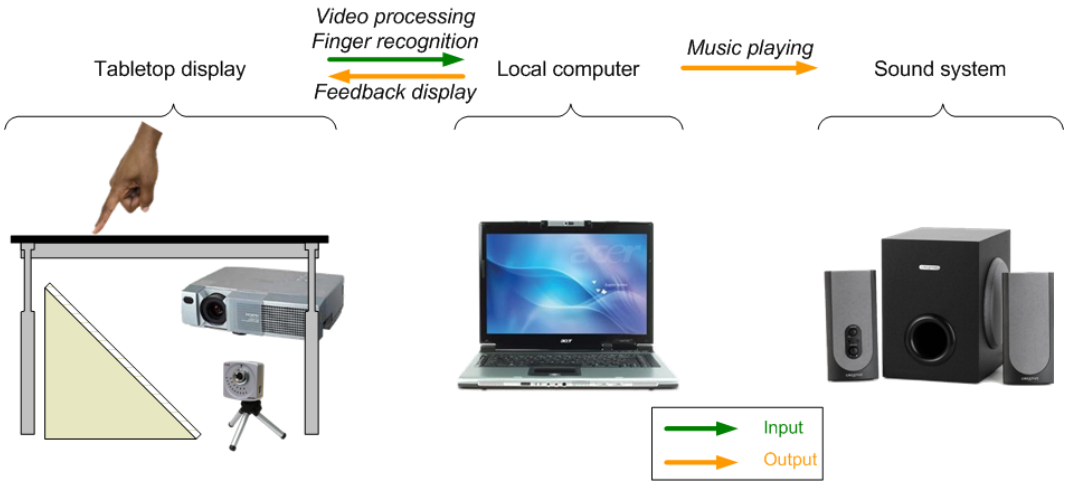


Figure 4.1: The three physical components and their interaction

This physical approach can be extended to offer a more precise view of demeTouch's system, putting forward the three conceptual components introduced earlier: the tabletop display (extended to the physical setup), the application, and the video processing engine. Figure 4.2 illustrates this view of the system.

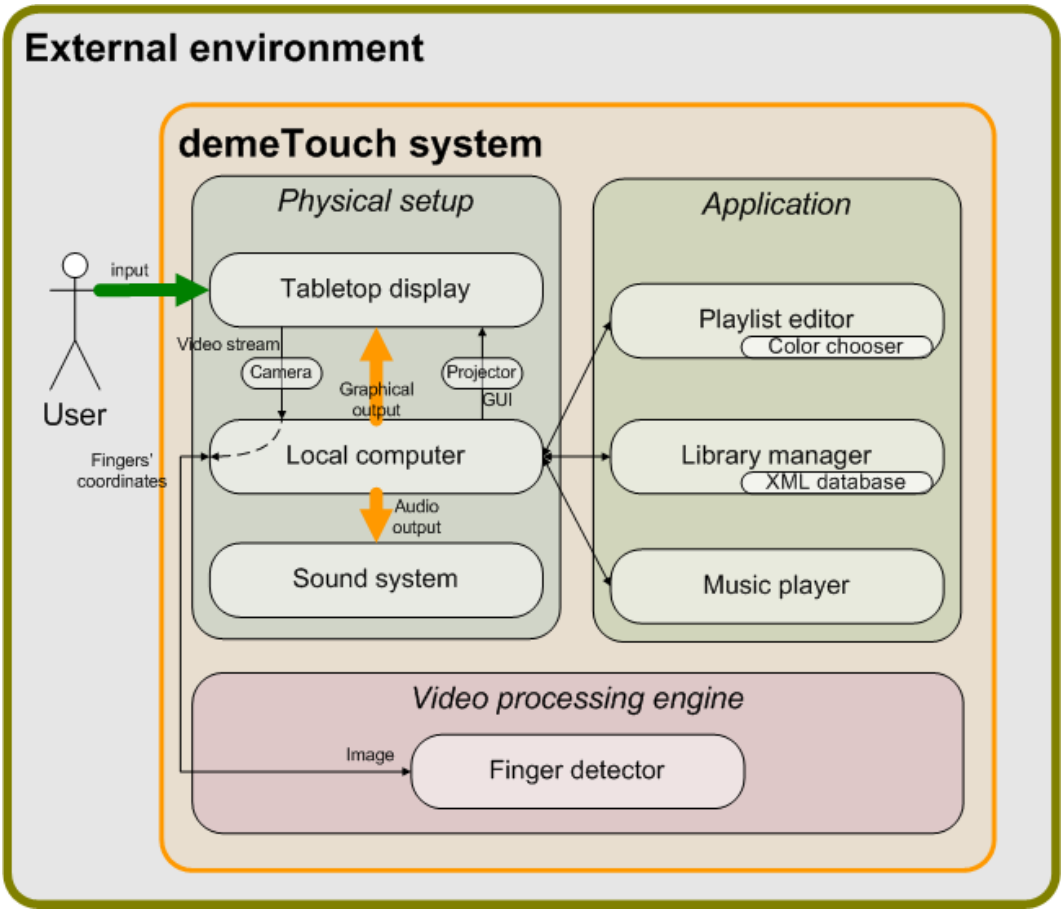


Figure 4.2: The three subsystems involved in demeTouch

From this perspective, the system can be explained as follows: the user interacts with it through the graphical user interface (GUI) displayed on the table's surface. The surface is filmed by the video camera which sends to the computer the raw image of the surface, with the images of eventual fingers touching the surface. Then these images are processed by the video processing engine in charge of extracting the finger's coordinates and sending them back to the computer. For every detected finger, depending on their location, the application interprets the user's command and modify the GUI, which is displayed on the table's surface via the projector. One of the challenges is to make all these processes communicating together efficiently. Indeed, they have to run in parallel: whereas the camera continuously films the table's surface, the graphical interface is not modified every time an image is sent to the computer.

4.2 The tabletop display

4.2.1 The required modifications

The setup used for this project has not been built especially for this occasion. Although it suits some of the requirements, a few modifications have been done to adapt its use to the present problem. As introduced in chapter 2, the original tabletop display is a wooden $120cm \times 171cm$ piece of furniture including a $100cm \times 75cm$ surface characterized by one side being semi-circular. The surface is made of a glass plate covered by a film dedicated to back projection. Video can be displayed on it thanks to a built-in combination of a wide-angle projector and a mirror. The mirror is used to increase the distance required between the projector and the table's surface without increasing the table's height. In its original state, the setup provides a solution for the system's visual output. Implementing the input solution consisted in adding a camera to the system that is set in a position that allow filming the whole table's surface avoiding an excessive image distortion.

4.2.2 The choice of a camera

The first idea was to use a digital video camera connected to a computer via a frame grabber. The main advantages of this solution is that the lens can be easily adapted to the required specification (in the present case a wide angle), and the high quality of the video stream in term of frame rate. The main disadvantage being the cost of such a solution: if lenses are accessible, cameras and frame grabbers are rather expensive, the total cost of the solution exceeding hundreds of dollars. In addition, most frame grabbers use a proprietary Software Development Kit (SDK) which requires to use a non-standard Application Program Interface (API). To conclude, using such a solution would require to develop a bridge between the proprietary API and standard drivers, which would make the system dependent on the camera used (unless such a bridge is developed for other devices).

Another solution consisted in considering web-cameras. Easily accessible for cheap, most of them include an internal frame grabber and offer a standard connectivity (USB or FireWire). The disadvantage of such a solution being the quality of the image retrieved. Guided by the reacTable project's team, the first experiments of a web-camera based video acquisition were done using a Philips Toucam Pro II™, which offers a good image quality (30 frames per second (fps) in 640×480 resolution) but suffers from a rather low-speed connectivity (USB 1.0). Indeed, transmission speed is very important as the camera is supposed to send continuously 640×480 images in 24bits color,

which corresponds to a $7,372,800 bits$ ($7.4 Mbits$) block of data per frame. Whereas USB 1.0 does not exceed a $12 Mbps$ transfer rate, USB 2.0 theoretical transfer rate is up to $480 Mbps$. To give the reader a better signification of these figures, while USB 1.0 would transfer the previous image in $614.4 ms$, USB 2.0 would transfer it in $15.36 ms$. Unfortunately, most USB 2.0 devices are far from these theoretical speed, and it is usually considered that they operate at $24 Mbps$. Finally, the best compromise between image quality, transfer speed and price was found in the Unibrain Fire-i™ digital camera, shown in Figure 4.3. This camera offers a $30 fps$ frame rate in 640×480 resolution, with a theoretical $400 Mbps$ transfer rate (IEEE 1394a-s400). Another advantage of this product is the possibility of using different lenses that are for example infrared light sensitive. Next section explains the importance of such a characteristic.

Figure 4.3: Fire-i™ digital camera by Unibrain

4.2.3 The need for infrared

The table's surface is used both for acquiring the position of eventual fingers in contact with it, and for displaying the graphical interface. Filming such a surface implies filming also what is displayed on it, which add a considerable amount of noise to the image, increasing the video processing's complexity. To avoid this unnecessary information in the image, a simple solution consists in switching the camera's color domain from visible light to infrared light. As the information displayed on the table's surface is in the visible light domain, by acquiring the infrared signal only, the camera is not affected by this noise. To increase the amount of infrared light reflected by objects, a source of infrared light should be added to the system. Figure 4.4 describes the final setup of the tabletop display. The infrared light source consists of two $100 W$ bulbs. Unfortunately, they also diffuse a small amount of visible red light, which justify the use of red light filters to avoid the table's surface being colored in red.

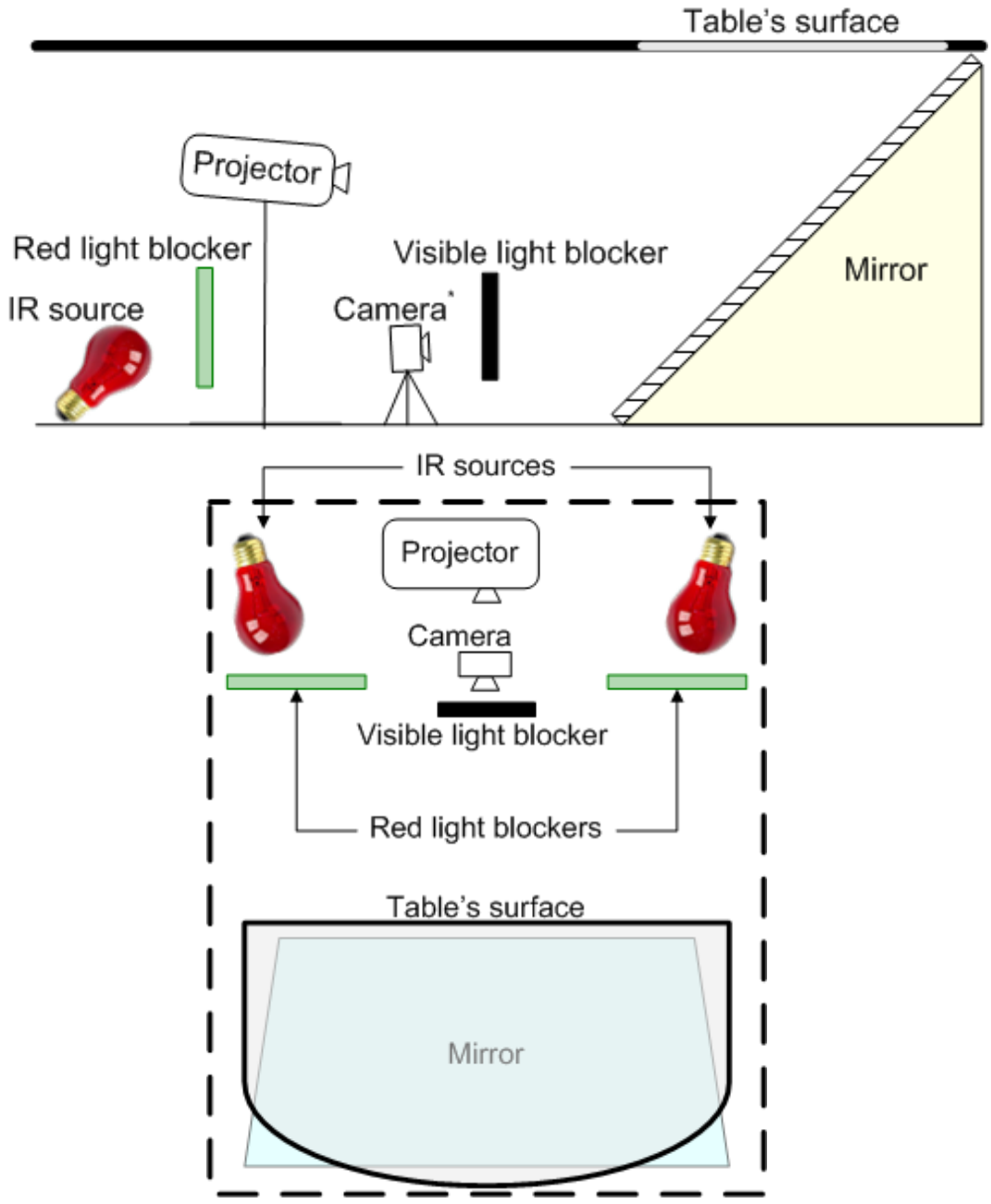


Figure 4.4: Final setup of the tabletop display with infrared system, from the side (*top image*) and from above (*bottom image*). (*) The camera is infrared light sensitive

4.3 The video processing engine

4.3.1 Creating the optimal combination of processing tools

According to [CP95], image processing is a succession of three phases:

- first, the **extraction** phase consists in looking for the desired features in the image. To do so, we distinguish the *preprocessing* operation from the *processing*. While the first operation is used to prepare the image to be processed, by enhancing some image's characteristics (such as the contrast) or removing the noise, the second operation consists in applying the proper algorithms that will result in putting forward the information looked for.

- then, the **treatment** phase consists in applying filters to the extracted information so that these information correspond exactly to what is expected.

- finally, the **interpretation** phase consists in replacing the results of the previous phases in the general context, confronting them to the original expectations. This phase is especially useful when the observed results does not match the expectations to understand the problem's source. Then, the preprocessing or processing phases can be modified. Otherwise, the results can be used as desired.

Considering the methods introduced in the previous sections, the brightness enhancement, subtraction and binarization processes represent the *extraction* phase, while the *treatment* phase is formed by the dilatation, spot creation and size discard processes. The *interpretation* phase is let either to the user when she adjusts the processing parameters or to the application when it interprets the finger positions. Figure 4.5 sums up this processing scheme.

Some algorithms are combined together to optimize the processing time. For example, when the subtraction is performed, pixels from the current image that differ from the reference image can directly be set to 255, thus the binarization is also performed without going through the array of pixels again.

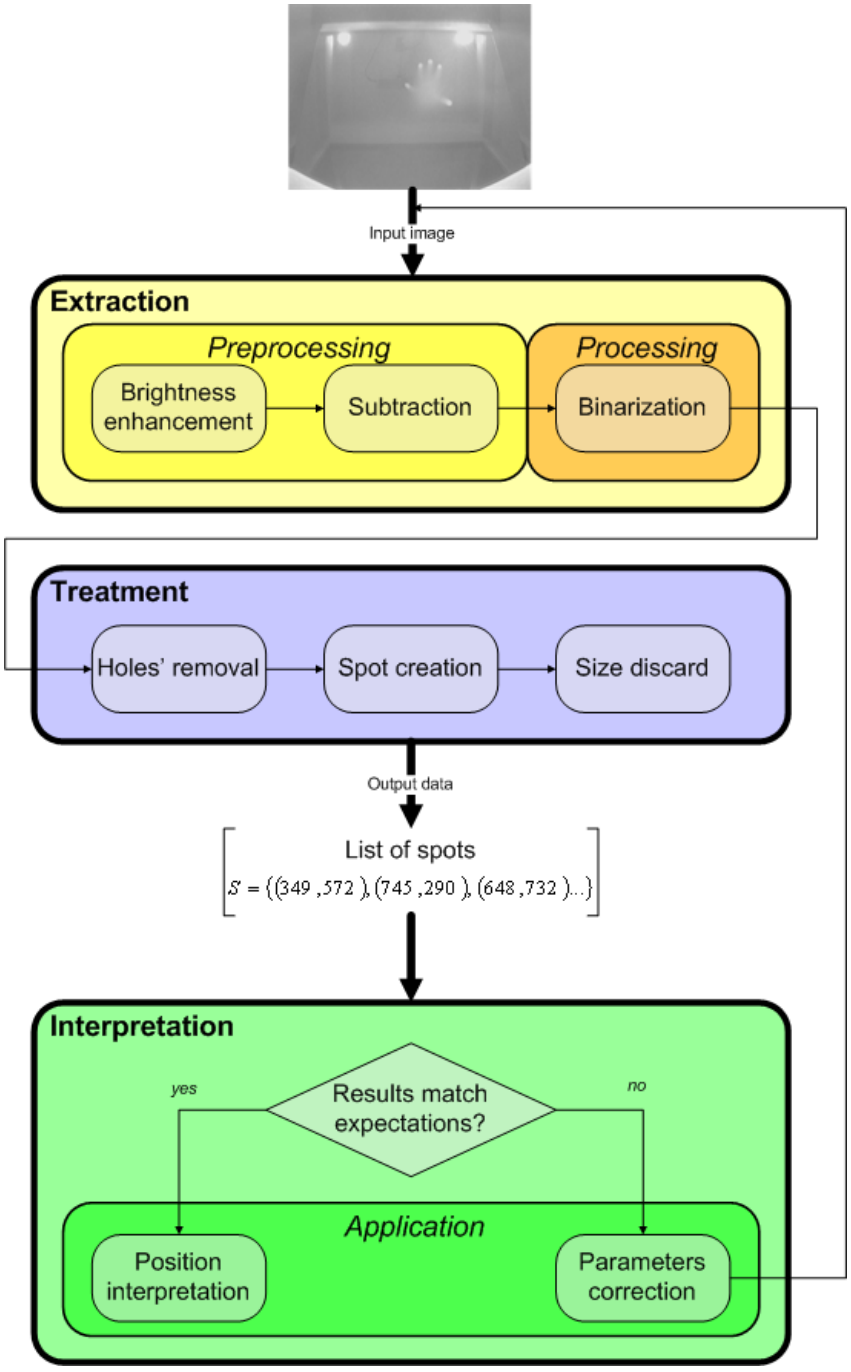


Figure 4.5: From the original image to the use of the fingers' coordinates

4.3.2 Sequencing the processes to extract fingers

This section uses the image processing tools defined in chapter 3, combine them and apply them to an example of image retrieved from the video stream provided by the

camera (Figure 4.6).

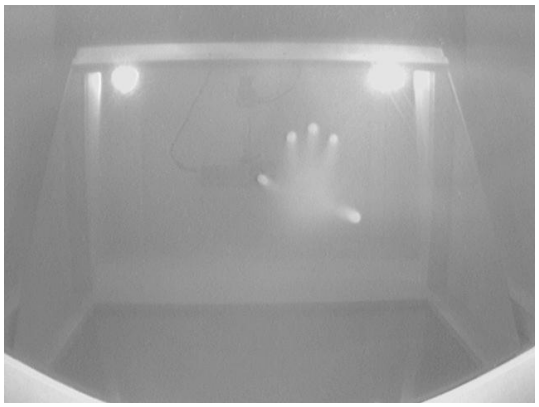


Figure 4.6: The current image to which the finger extraction algorithms are applied

Brightness enhancement

First, the pixel values are normalized in 40×30 pixels windows (Figure 4.7).

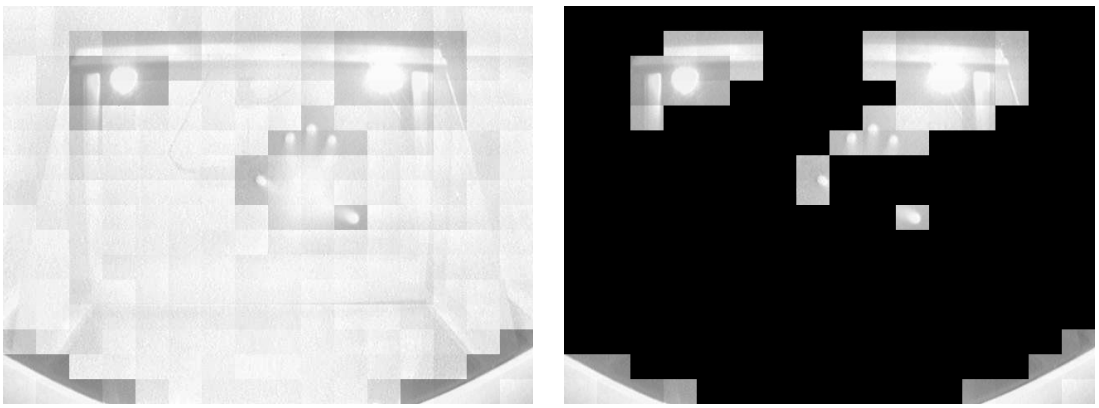


Figure 4.7: Brightness enhancement illustration: the current image's brightness is enhanced in every windows first without thresholding the multiplication factor (*left image*). Then the multiplication factor's threshold $\varepsilon_{Br} = 1.3$ is introduced (*right image*): only windows containing bright spots are kept from the current image

Subtraction and binarization

Then, the current image is deducted from the reference image. In the same time pixels differing from the reference image are set to 255. As depicted in Figure 4.8 the subtraction operation suits well a noisy environment and manages to extract the spots corresponding to fingers. This is mainly due to the high contrast in the area showing fingers. Thanks to that characteristic, the ε_S threshold can be set to a high value, discarding every small variations between the current image and the reference. In this example, $\varepsilon_S = 26$.

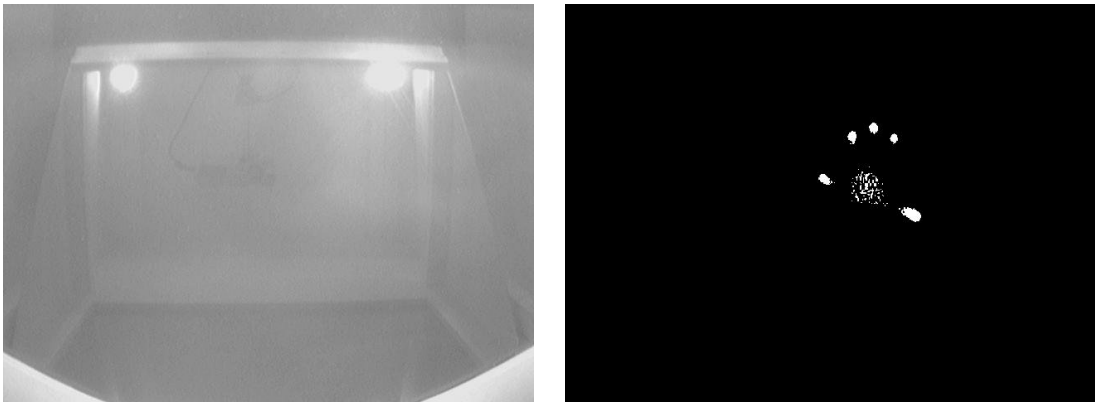


Figure 4.8: Subtraction including binarization example: the current image's pixels (*right image*) that differ from the reference image's ones (*left image*) are set to 255

Holes' removal

If most of the noise has been removed to leave the finger-like areas only, some of them present irregularities and/or are too small to be fingers. Applying the holes' removal algorithm discards these noisy areas and clarifies the real fingers, as depicted in Figure 4.9 where the areas are 4×4 pixels big.

Spots creation and size discard

Applying the previous algorithms results in the extraction of a list of spots that match the characteristics of fingers in term of image analysis. Nevertheless, according to 4.9,

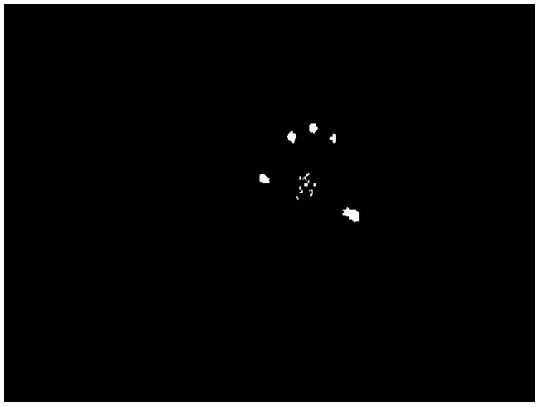


Figure 4.9: Holes' removal illustration: noisy spots are removed from the binarized image and fingers are clarified

most of these spots comes from the noise introduced by the user's hand palm. In this example, the list of 14 spots is filtered to keep only spots with a finger-like size. Here, spots are considered as fingers only if their size is ranged from 55-220 pixels. According to these parameters, the final list of finger coordinates extracted from the original image (Figure 4.6) is listed below (from top to bottom), and illustrated by Figure 4.10.

$$\{725,300\} \quad \{682,321\} \quad \{763,323\} \quad \{628,416\} \quad \{800,499\}$$

Figure 4.10: Size discard illustration: only the spots that match a certain size are considered as fingers

4.3.3 Calibration issue

Active area

As introduced in the brightness enhancement process' description, the infrared repartition on the table's surface is not homogeneous. As a result the finger recognition rate differs from zone to zone, even after applying optimization algorithms. To handle this issue, a simple solution consists in using as an active area only the zone of the table where the recognition rate is high, using other zones for display purpose only. As depicted in Figure 4.11, the area showing the highest finger recognition rate is located in the center of the table's surface. In addition, considering the user standing in a normal position in front of the table, the dimensions of the surface would make the furthest areas from the user difficult to reach (see Figure 4.12).

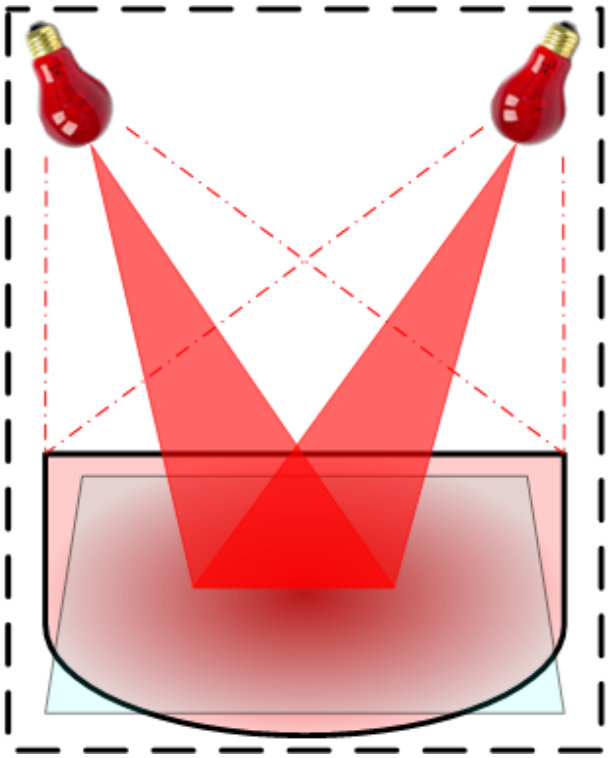


Figure 4.11: Active area illustration: the center of the surface (dark red zone) offers the highest level of infrared light

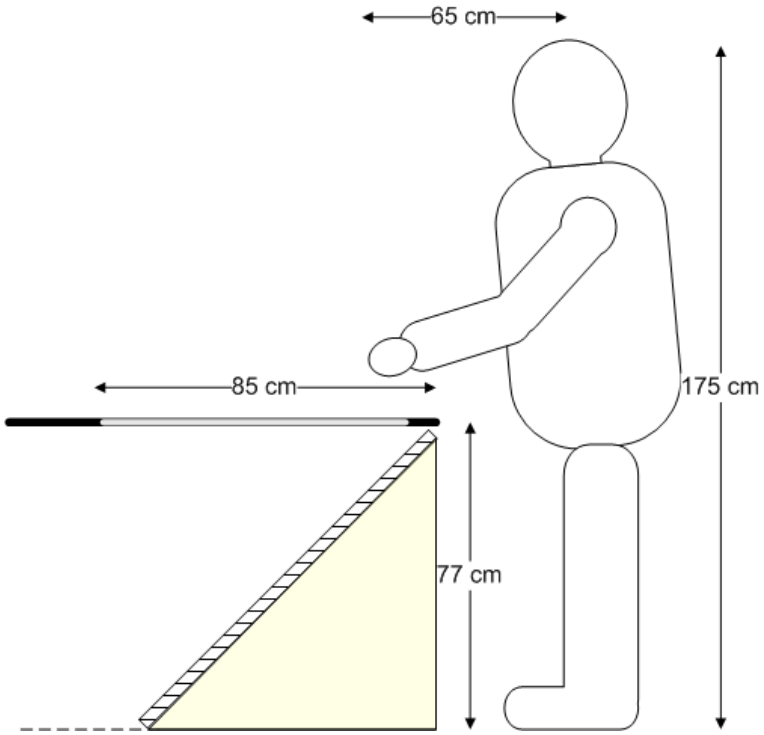


Figure 4.12: Table dimension issue: for a user standing in a natural position, the furthest surface's areas are difficult to reach

Camera's position

As the horizontal and vertical distortions depend mostly on the physical setup, assuming that the projector is placed in the ideal location, the closer the camera to this location, the smaller the distortion and as a result the smaller difference between the expected values and the measurements. Testing the finger recognition after calibration requires first to optimize the camera's location, before computing the equations that allow to switch from the camera's coordinates system to the projector's ones.

First, Figure 4.13 shows the impact of a poor camera's positioning. It illustrates the camera being turned to the left side of the table's surface (viewed from the camera's location), resulting in an important difference in the x-axis's position on the right side of the table's surface viewed from the user point of view.

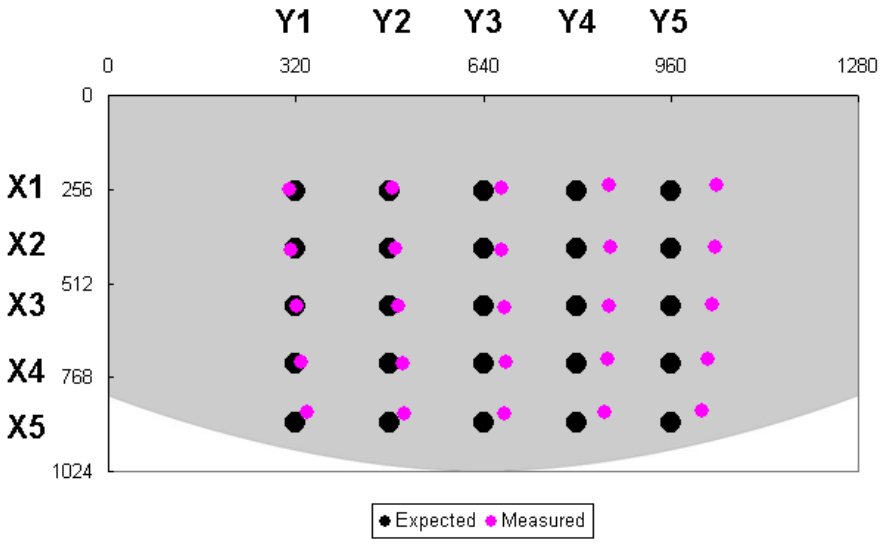


Figure 4.13: An example of poor camera's positioning

In comparison, Figure 4.14 illustrates an optimized camera's positioning: the central point (the point that is detected with the most accuracy without calibration) is located in the center of the table's surface, instead of the very left edge of the area in the poor positioning example. This optimization limits the distortion effect and simplifies the calibration issue.

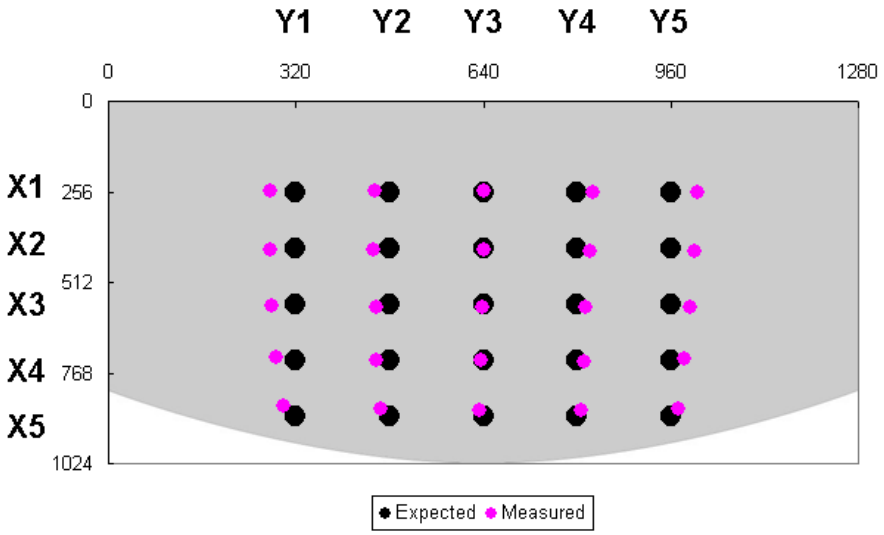


Figure 4.14: An example of optimized camera's positioning

Considering this configuration as the optimal that can be achieved by simply moving the camera by hand, Figure 4.15 illustrates the differences in the coordinates along the horizontal lines (top image) and the vertical lines (bottom image). The difference in scale between the two types of distortion informs about the importance of each distortion: the error introduced in the x-axis is clearly more important than the one introduced in the y-axis.

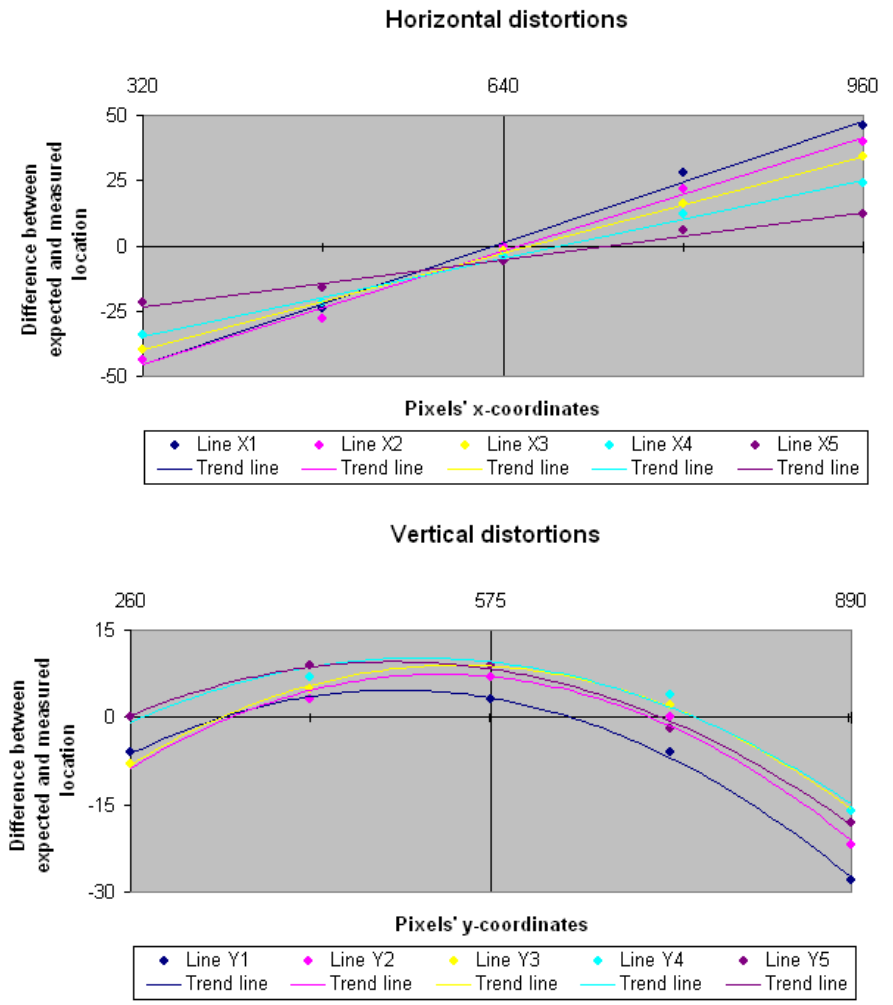


Figure 4.15: Distortions representation: the *top image* illustrates the distortion along the x-axis, while the *bottom image* illustrates the distortion along the y-axis

Switching from the camera's coordinates to the projector's ones

As discussed at the end of chapter 3, a simplified solution to the calibration issue is implemented in demeTouch. Considering the previous remark concerning the active area, the mosaic of twenty five spots depicted in Figure 4.16 is displayed in the center of the table's surface. The user is asked to place a finger successively on each spot, and its location in the camera's coordinates system is recorded.

Then, when a finger is detected, the vertical and horizontal functions that correspond to the closest lines to the finger are applied to its coordinates in the camera's system, resulting in the corresponding coordinates in the projector's system.

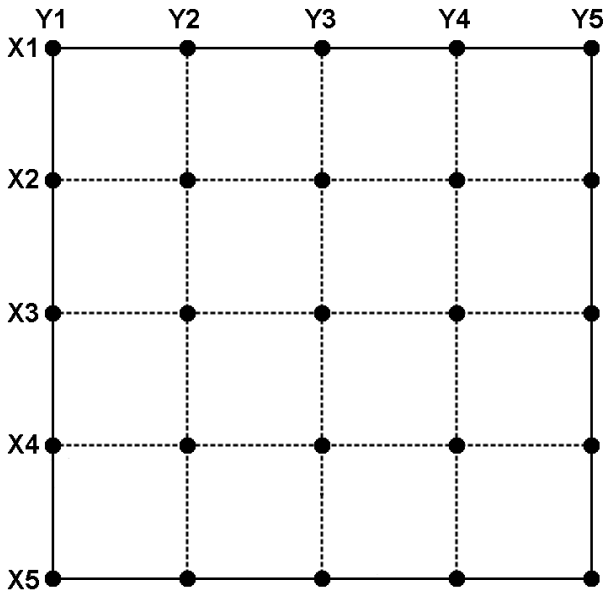


Figure 4.16: Calibration with square mosaic: the coordinates in the camera's system are recorded and confronted to the known values in the projector's coordinates system

In the optimal configuration introduced in the previous section, a linear approximation is sufficient for correcting the x-coordinates, while a second order regression is required for the y-axis. The set of equations computed in that configuration is provided in table 4.1. Finally, Figure 4.17 shows the result of the correction using these equations.

Horizontal distortion		Vertical distortion	
Line	*Equation*	*Line*	*Equation*
X1	$\delta x = 145.0E^{-3}x - 91.6$	Y1	$\delta y = -2.0E^{-4}y^2 + 20.1E^{-2}y - 44.9$
X2	$\delta x = 136.3E^{-3}x - 89.2$	Y2	$\delta y = -2.2E^{-4}y^2 + 23.5E^{-2}y - 55.0$
X3	$\delta x = 116.3E^{-3}x - 77.2$	Y3	$\delta y = -2.1E^{-4}y^2 + 23.0E^{-2}y - 53.7$
X4	$\delta x = 93.8E^{-3}x - 64.8$	Y4	$\delta y = -1.8E^{-4}y^2 + 18.0E^{-2}y - 35.8$
X5	$\delta x = 56.3E^{-3}x - 41.2$	Y5	$\delta y = -1.8E^{-4}y^2 + 17.2E^{-2}y - 32.6$

Table 4.1: Set of equations used for calibration in ideal conditions

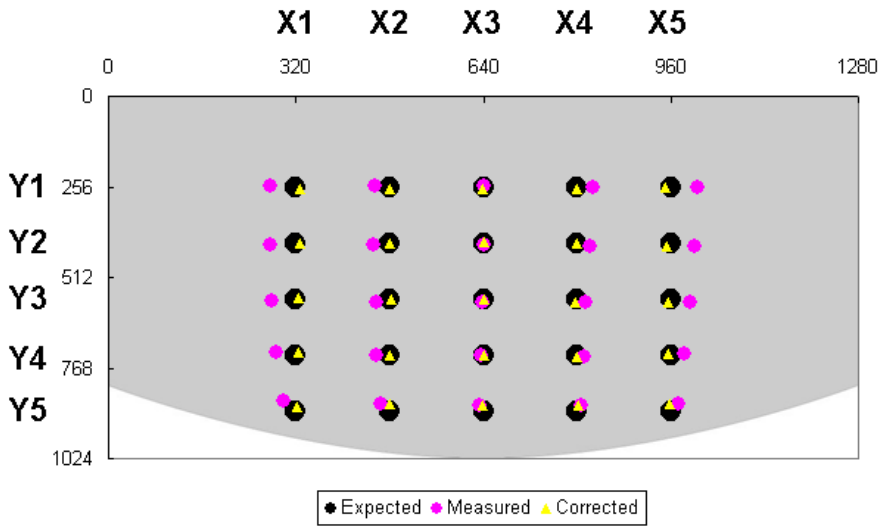


Figure 4.17: Corrected coordinates in ideal configuration

General calibration solution

The previous equations consider the physical setup to be optimal. In reality, to assure the best recognition rate without going thought the calibration procedure every time the application is launched, it is considered that the setup might slightly change (the table or the camera can be moved), resulting into finger location imprecisions. To avoid this situation, a more general solution is implemented in the image processing engine. Instead of linear and quadratic equations, fourth degree polynomials are used to correct the detected coordinates. Tables 4.2 and 4.3 give this set of equations, while Figure 4.18 illustrates the accuracy of this correction applied to the optimal physical setup.

Horizontal distortion	
Line	*Equation*
X1	$\delta x = -8.9E^{-10}x^4 + 2.0E^{-6}x^3 - 1.5E^{-3}x^2 + 0.6x - 142.0$
X2	$\delta x = 1.3E^{-9}x^4 - 3.6E^{-6}x^3 + 3.6E^{-3}x^2 - 1.4x + 132.0$
X3	$\delta x = 3.8E^{-10}x^4 - 1.0E^{-6}x^3 + 9.7E{-}4x^2 - 0.3x - 24.0$
X4	$\delta x = 3.8E^{-10}x^4 - 1.2E^{-6}x^3 + 1.3E^{-3}x^2 - 0.5x + 22.0$
X5	$\delta x = -3.8E^{-10}x^4 + 7.7E^{-7}x^3 - 5.0E^{-4}x^2 + 0.2x - 44.0$

Table 4.2: General equations implemented in the fingers detector engine (X-axis)

Vertical distortion	
Line	*Equation*
Y1	$\delta y = -2.6E^{-10}y^4 + 5.2E^{-7}y^3 - 5.5E{-}4y^2 + 0.3y - 52.6$
Y2	$\delta y = 4.9E^{-12}y^4 - 1.8E^{-7}y^3 + 7.7E^{-5}y^2 + 0.1y - 30.8$
Y3	$\delta y = 1.4E^{-10}y^4 - 3.6E^{-7}y^3 + 1.2E^{-4}y^2 + 0.1y - 37.8$
Y4	$\delta y = -4.0E^{-10}y^4 + 7.1E^{-7}y^3 - 5.6E^{-4}y^2 + 0.2y - 35.6$
Y5	$\delta y = 5.5E^{-10}y^4 - 1.2E^{-6}y^3 + 7.0E^{-4}y^2 - 0.1y - 4.2$

Table 4.3: General equations implemented in the fingers detector engine (Y-axis)

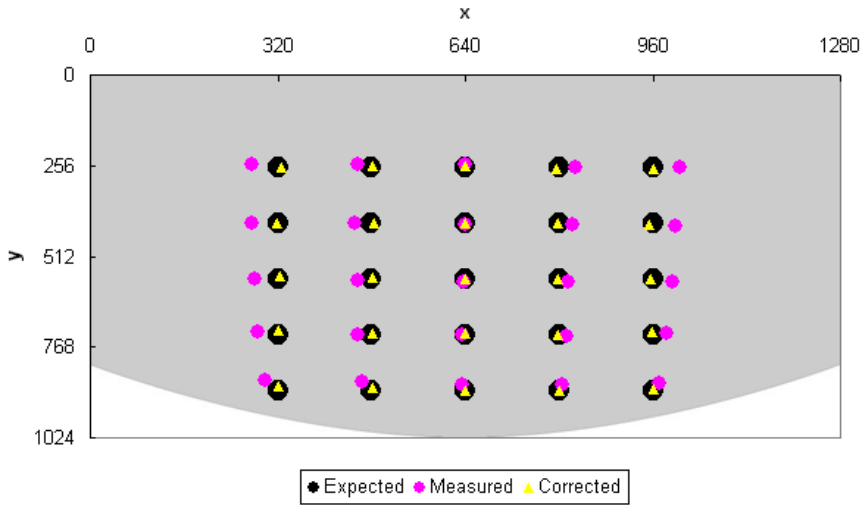


Figure 4.18: Corrected coordinates using general solution in ideal configuration

4.4 The music application

4.4.1 General presentation

As introduced in chapter 2 when presenting the demelo project, demeTouch's music application is largely inspired by demelo's graphical user interface. However, it is adapted to suit the tabletop display's characteristics. It implements the following elements: a music player, a color-based playlist editor and a library manager. Designing the music application requires to take into consideration the challenges related to the physical setup and the finger detector engine. In addition, the application has to respect the basic requirements introduced in chapter 3 to offer the user a nice interface which is intuitive to use and navigate through. Finally, it has to include basic standard functionalities usually proposed by similar applications to give the user an impression of familiarity when using it.

An active area and a display area

As introduced in the calibration design section and illustrated in Figure 4.19, the table's surface is divided into two types of zones:

- the **active area**, located in the lower center part of the surface, is the area where the user can interact with the application using her fingers.
- the **display area**, surrounding the active area, is used to provide visual feedback to the user, extra static information that does not require to be manipulated.

The focus will be first put on developing the active area, as it is the most challenging one. Then, the display area will be eventually used to provide extra information to the user.

One menu and three different panels

Due to the active area's dimensions and in order to leave the display clear and readable, all the interactive information cannot be displayed at the same time on the table. Thus the active area is split into a main panel which takes most of the space, and a menu located below the main panel. Figure 4.20 illustrates this idea by showing the menu formed by three buttons which allow to switch from an interface to another.

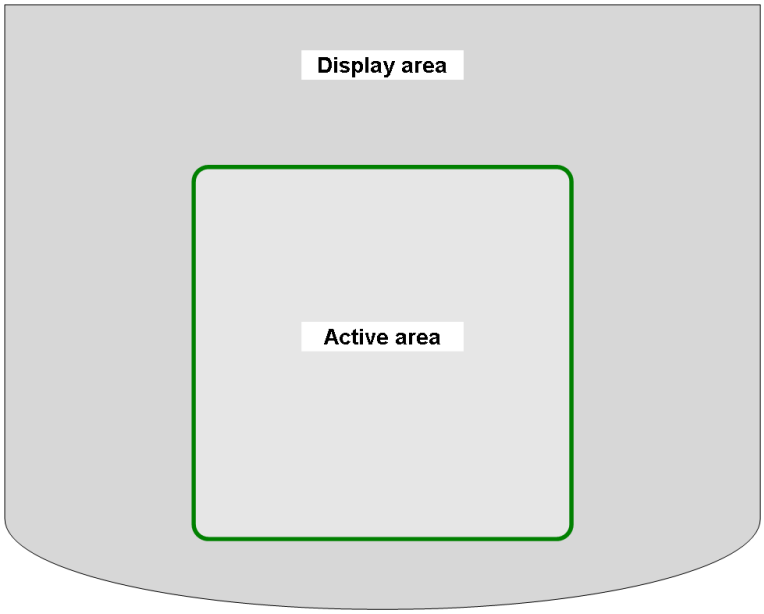


Figure 4.19: The table contains an active area and a display area

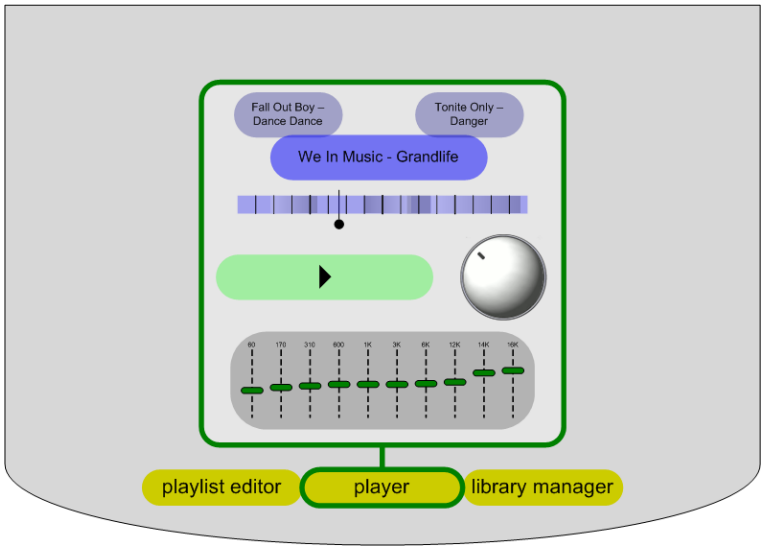


Figure 4.20: The three buttons menu allows to switch from an interface to another

The importance of the visual elements' size

The size of graphical elements is of a big importance for two reasons: first, big elements are preferred to small ones for readability reasons: the projector's resolution does not allow to clearly display tiny components, which would be difficult to see anyway due to

the configuration of the table. Another problem would appear when it comes to display text: users would have to bend themselves to read, making the application inconvenient. A solution could be to display small visual elements on the closest area from the user and increase the size of further elements.

Secondly, bigger elements prevent from misinterpreting finger positions. Clicking with a finger on a big button is much easier than doing the same on a very small component. In addition, if the system often misunderstands fingers' position, it will inevitably lead to the user's frustration. All these remarks considered, big visual elements are preferred in demeTouch's design.

4.4.2 The panels in detail

The *Player* panel

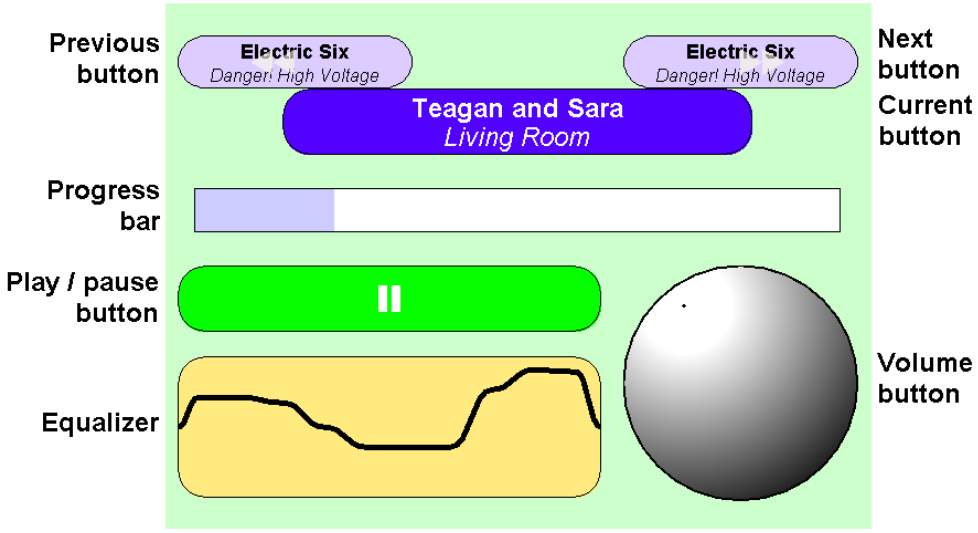


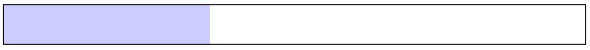
Figure 4.21: The Player panel offers basic functionalities to play the music

As depicted in Figure 4.21 this panel proposes the basic functions of a music player: a play/pause button, a set of three buttons (previous, current and next) to navigate through the playlist, a progress bar, a volume button and an equalizer.

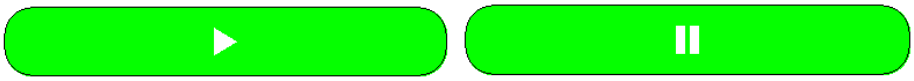
The interface is simple for two reasons: first, building a music player with many functionalities is not the main goal of the project, extra features can eventually be added later to the player if needed. Secondly, the core idea behind the application is to simplify the access to the user's audio library. Thus simplicity and ease are also required for the application's design.

Playlist navigator These tree buttons allow to play from the beginning either the previous, the next or the current title of the playlist. The corresponding artist's name and song's title are displayed on each button. Once clicked on the previous or the next button, the three panels are updated to always display the current artist and title on the center button.

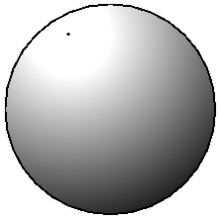
Progress bar This component shows the elapsed and remaining time on the current title. Clicking on the component will adjust the playing position to the selection.



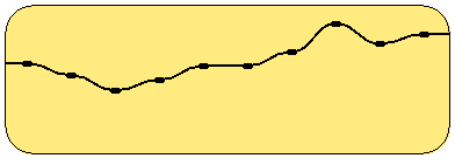
Play / pause button This button switches from one state to the other when clicked, the play button being displayed when the song is paused, and the pause button being displayed when the song is playing.



Volume button This button allows to control the audio signal gain from 0.0 - 1.0. Turning the button to the left decreases the volume, while turning it to the right increases it.



Equalizer This component allows to adjust the pitch of the song being played by either amplifying or reducing the level of a set of determined frequencies. The component's left side represents the low-pitched frequencies, the high-pitched frequencies being represented on the right side.



The *Playlist editor* panel

This panel allows the user to create playlists by manipulating colors. Before presenting the panel's components illustrated in 4.22, some explanation are provided on the design's choices in terms of color space model and representation.

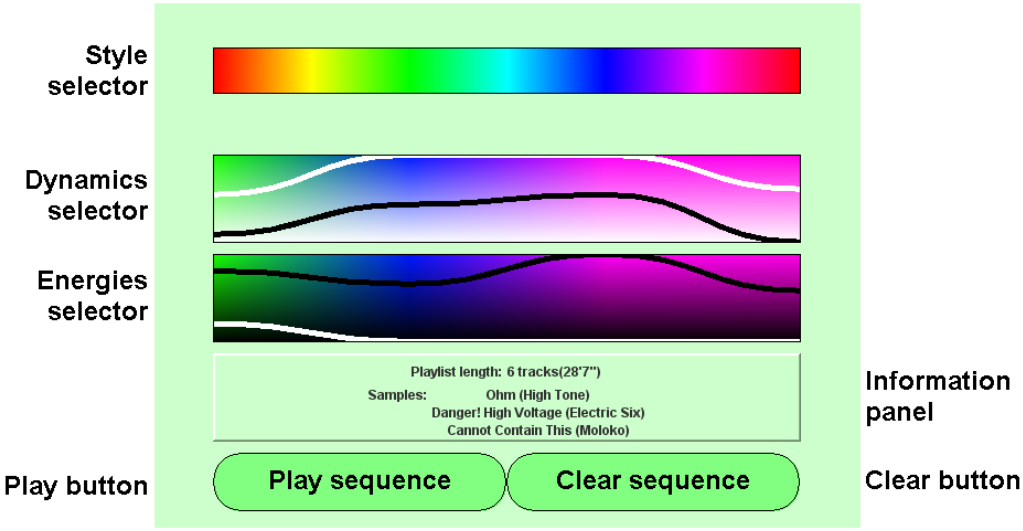


Figure 4.22: The playlist editor allows the user to create a musical sequence

Color model demeTouch considers the colors in the Hue-Saturation-Value model (HSV), each component being described as follows, and illustrated in Figure 4.23.

- the **hue** defines the color (red, yellow...) ranged from 0 to 360.

- the **saturation** corresponds to the color's degree of purity. Ranged from 0 to 100%, the more saturation the clearest the color, whereas the less saturation the more faded the color.

- the **value** influences the color's brightness. Also ranged from 0 to 100%, the less brightness the darker the color.

Two other famous color spaces are:

- **RGB** stands for Red, Green and Blue. This color space is called *additive color mixing*: colors are formed by the addition of the three components. It is commonly used in television and in computer graphics.

- **CMYK** in comparison, is a *subtractive color mixing* as levels of cyan, magenta, yellow and black are subtracted from white to create the color. It is mainly used in printing domains.

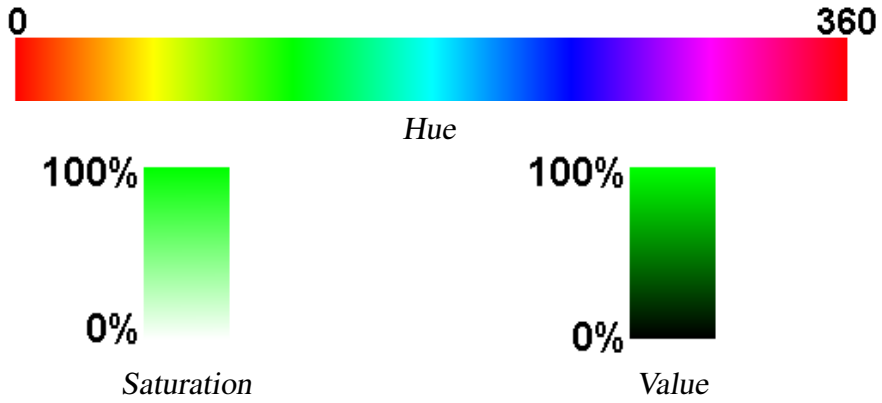
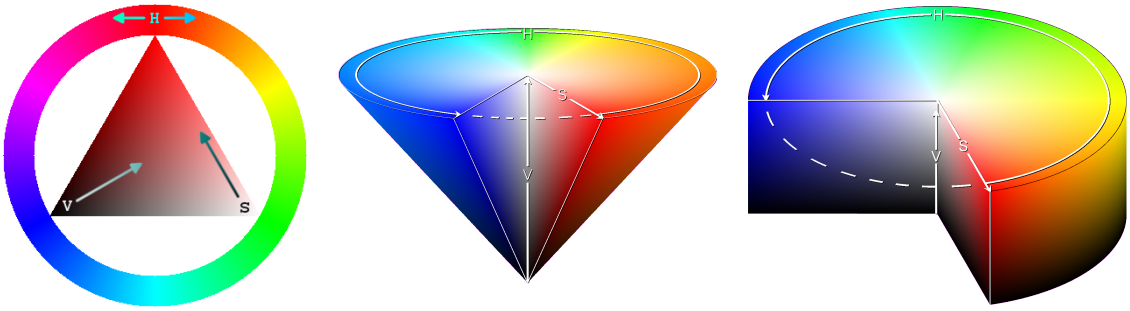


Figure 4.23: The HSV model's three parameters. Saturation and value are shown for hue=120

Choosing the color selector The difficulty introduced by such a color model comes with its graphical representation. Figure 4.24 shows three types of representations that cover the full range of colors in the HSV space. Out of these three visual representations the triangular one would be the easiest solution to pick up a color from, without introducing a third dimension to the tabletop display. Unfortunately, this visual representation introduces two other problems when it comes to apply it to demeTouch.

First, the visual aspect might seem a bit confusing, especially when manipulating the saturation and value, which tend here to be mixed together, making unclear the role each of them plays in the final color. Secondly, this tool would provide an excellent accuracy for picking one color, but would be more complicated when selecting a whole sequence of colors. For example, how to select a *yellow-blue-red* sequence without considering the intermediate colors?



Images from Wikipedia, the free encyclopedia (article HSV color space)

Figure 4.24: Three visual representations of the HSV space: triangular (*left image*), conical (*center image*) and cylindrical (*right image*)

demeTouch's colors selector The HSV's visual representation used in demTouch clearly distinguishes the three components which combined together form the final color. Three separate bars are used to display each of the three parameters as described in Figure 4.25. The top bar is used to first select the color sequence. Every time a color is selected, it is added to the two other bars which represent the saturation and value levels.

From colors to music The music-color association used in demeTouch reuse the concept developed for demelo, and works as follows.

- the color **hue** is mapped to the music **style**, according to Figure 4.26. This scale can be discussed as it introduces subjective information, so it should be considered as a general tendency more than an exact matching table.

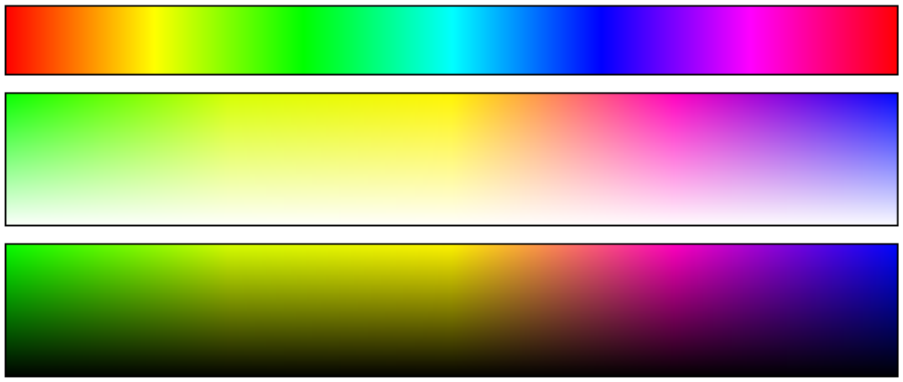


Figure 4.25: demeTouch's color chooser: the top bar illustrates the colors, the middle bar the saturation and the bottom bar the value

- the color **saturation** corresponds to the **dynamics**, which is the evolution in tempo and variations in the musical scale over the piece of music. Not to be mixed up with the energy, as a piano solo can offer a lot of variations in the music composition's structure, while a song full of instruments can be flat in term of scale variations.

- the color **value** corresponds to the music **energy**: the brighter the more punchy, while the darker the quieter.

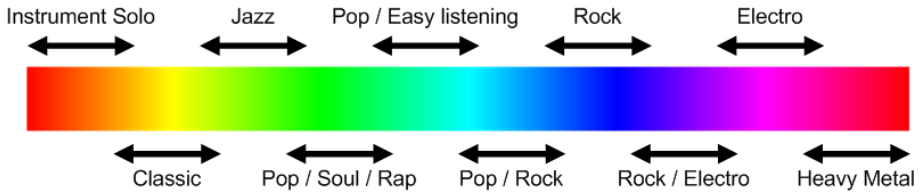


Figure 4.26: Music-color association scale

A parallel with the color-mood association Considering the Morton Walker's color-mood association scale introduced in chapter 3, a direct parallel can be done for some colors, while others might seem a bit more obscure. For example it is easy to associate the red color of heavy metal with passion, for the violence that both the music and the sentiment embed. The same direct link can be seen between classical music and attention, represented in orange. If the same kind of association can be easily made for other colors (blue: rock and strength or gray: low dynamic, medium energy and

neutral), these are still subjective and can be discussed. In addition, it seems more difficult to associate colors such as pink or yellow with weakness and stress. However, these parallels illustrates the relation between colors, music and mood, even if it has to be adjusted to the user.

Creating a music sequence To create a playlist using demeTouch's color selector, the user has first to select the sequence of colors that will be played in order from left to right. Then, by moving the lines on the saturation and value bars, she can define the desired ranges of dynamics and energy. For example she can decide to listen to calm and quiet music (low saturation and value) at the beginning of the playlist, and then switch to richer and more punchy music (high saturation and value). Figure 4.27 illustrates this example for a jazz-electro/pop-Rock sequence (green-magenta-blue).

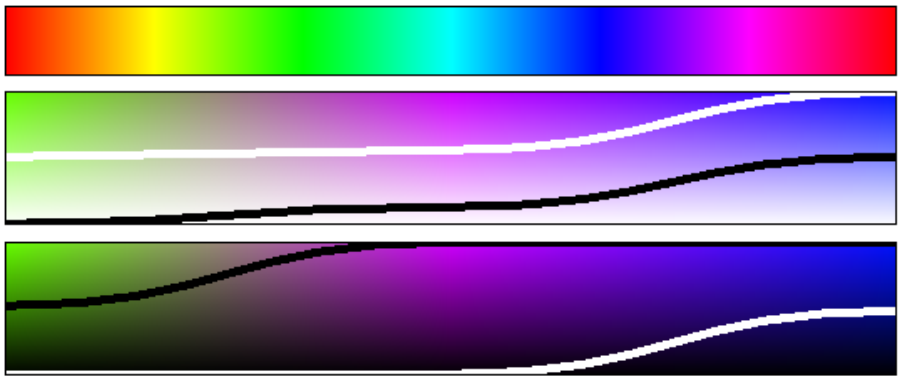
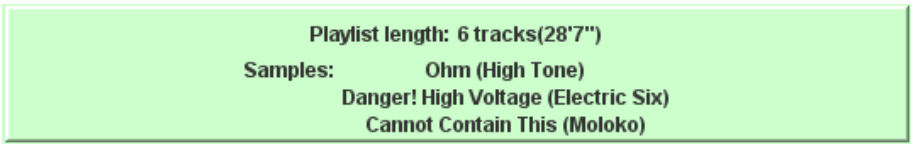


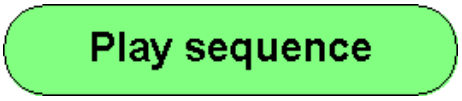
Figure 4.27: An example of playlist that will play first calm and quiet jazz, then medium electro/pop and finally rich and punchy rock

Controlling the playlist While the sequence of colors is being selected and the ranges of dynamics and energy adjusted as desired, the user gets some information about the resulting playlist in the *Playlist Information* panel, located below the color selector bars. This information concerns the playlist length, its number of tracks, and up to three examples of title (and the corresponding artist) randomly picked up from the playlist. As the user cannot act on this panel, it could be moved to the display area around the active panel. However, it provides information directly in relation with the color sequence defined in the above color selector, which justifies its presence in the active area.

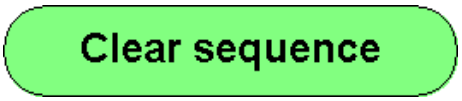


Once the playlist is created After the user has created her playlist, she can either start playing it or clear the sequence by clicking on the corresponding button.

Play sequence This button switches from the *Playlist editor panel* to the *Player panel* and starts playing the created playlist with the first title.



Clear sequence This button empties the sequence previously created, and initializes the panels, showing that a new sequence can be created.



The *Library manager* panel

This panel offers the user basic tools to manage her audio library.

Library panel This panel displays to the user her library's content: ordered by default according to the chronology they have been added to the library, the entries can also be ordered depending on their hue value, dynamics, energy, title, artist or album by clicking on the corresponding column header.

To navigate through the library, the user should use the up and down arrows located on the right side.

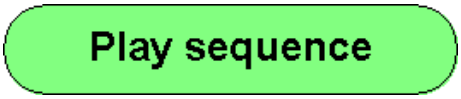
Figure 4.28: The library panel is used to manage the audio library

She can select and unselect entries by clicking on the corresponding row in the library content panel.

Unselected row

Selected row

Play sequence button This button allows the user to launch the playlist she created by manually selecting songs from the library panel. Once clicked, the panels switch to the *Player* one, and the playlist starts.



CHAPTER 5

Implementation

After presenting the system in chapter 4, this chapter presents some technical details concerning its implementation: what to use to upgrade the table, what language to use to develop both the finger detector and the application? These are the questions that this chapter answers.

5.1 Physical setup's details

5.1.1 Positioning and marking the physical elements

It has been stated that the position of each element forming the whole physical setup is extremely important. A poor organization of the under table space would result in a poor finger recognition rate, an inaccurate display on the table's surface, and would make the application unpleasant to use.

To handle this problem, and to avoid going through tedious calibration and repositioning of the physical elements, some simple rules had to be respected while developing a solution to the calibration issue. The table itself and all the elements under it are marked, either on the floor or on themselves, to fix their location according to others. Figure 5.1 gives the precise measurements that rule the setup under the table.

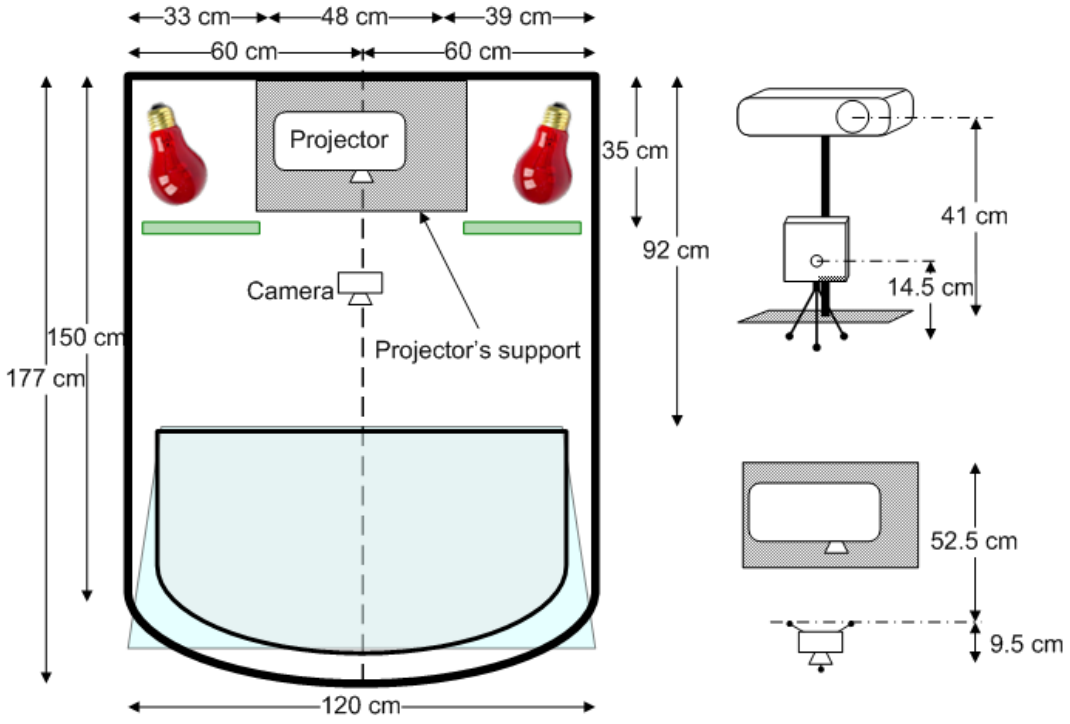


Figure 5.1: The precise measures which make the physical setup easy to repeat

5.1.2 Choosing affordable components

Except from the film covering the glass plate of the table's surface, all components included in the table setup are easy to find and rather cheap: the structure is entirely wooden-made, the projector is midrange easily accessible on the Internet. Thus, when it came to add components to the setup, it was natural to look for elements that match this low-price logic. The infrared-sensible camera used presented in chapter 2 was the best solution in term of price over quality ratio by the time of printing this document. In addition, the red bulbs used as infrared sources can be easily found in pet shops, as this type of material is mostly used to heat up vivariums. Finally, the two color filters used are simple sheets of green clear plastic.

5.2 Programming choices

5.2.1 Why using the Java™ programming language?

The choice of Java™ as the programming language has been guided by two main factors.

- Concerning the **finger recognition engine**, Frederic Boulet and the author agreed on this language first for their good knowledge of it. In addition, the Java™ Media Framework (JMF) provided by Sun Microsystems, Inc. offers an intuitive introduction to web cameras' manipulation. It allows to easily understand how to retrieve and process the video stream given by the camera.

- When it came to choose the language to develop the application, Java™ was a natural choice to integrate the finger detection engine, making the overall system as homogeneous as possible. Moreover, using Java for the whole system makes it portable, platform-independent. Finally, combining JMF with the Java™ Advanced Imaging (JAI) API (see [Inc99]) provides a rich set of tools to get the images from the camera, process them, extract useful information and display resulting graphical components in a smooth and built-in manner.

5.2.2 How XML defeated MySQL?

demeTouch's media library uses an XML file to store the information concerning the library entries: a file path, the three color features (hue, saturation, value), the track duration and the editorial information (title, artist, album). A natural choice for storing these data would be to use a regular MySQL database, using for example MySQL Connector/J that allows to converts Java™ Datadase Connectivity (JDBC) into MySQL language.

The two solutions have been tested independently under the following use case: a varying number of entries has to be added to the library. For both XML and MySQL solutions, the data to be inserted into the database is first formatted, then added to the database. After adding the whole entries, the file is saved on the disk. The time spent to do these tasks is measured for different amounts of entries to be added to the library. While the MySQL test uses MySQL Connector/J, the XML test uses JDOM, a library that allows to manipulate XML content in Java™. Whereas these tests have been conducted under three conditions of CPU usage (low, medium and high), the differences between them is not significant. Table 5.1 gives the results after conducting the tests

for a number of entries varying from 10-100,000. The three values in italic are linear estimations based on the previous results. These results prove the fastness of the XML version, making this technology the logical choice for demeTouch.

	Time (in ms)	
Number of entries	XML *(JDOM)*	MySQL *(MySQL Connector/J)*
10	16	250
50	31	1,625
100	31	2,938
500	47	13,844
1,000	63	33,094
5,000	156	167,188
10,000	313	*334,854*
50,000	1,422	*1,677,326*
100,000	2,985	*3,355,417*

Table 5.1: MySQL vs. XML test, illustrating the time (in ms) needed by both XML and MySQL to format a certain amount of entries, add them to the database and save the file

CHAPTER 6

Tests

This chapter details the way demeTouch performances have been evaluated. Two parts have been tested: first, the finger recognition engine, then the music-color association. For each of them, the testing procedure is first explained, then results are provided and commented. They give an objective view upon demeTouch performances from both the technical (finger recognition engine) and visual feedback (music application) perspective.

6.1 Finger recognition's evaluation

6.1.1 Test presentation

The goal of this test is to experience the finger recognition engine in a real environment. Eleven persons are asked to go through two exercises consisting in placing a certain number of fingers in different areas of the table's surface. Before detailing these exercises, it is important to note that the physical setup is not optimized, and the parameters used for the image processing are wide in order to suit the maximum types of fingers. In addition, no requirements are given to the test user: he is free to choose both the size of the finger areas in contact with the table and the pressure he uses. In addition, the test user is asked not to lean on the table, to keep the physical setup identical from the beginning to the end of the test. Finally, the test user does not see the control panel showing the finger detection results. Thus he cannot adapt the finger areas touching the table and/or the pressure used.

One finger per area

First the table's surface is divided into sixteen rectangular areas, a black spot being displayed in the center of each of them. The test user is asked to successively place one finger on every spot, without paying attention to the order in which the areas are considered. For each area the number of fingers detected and their location are recorded. If their location is important, it is more important to record the number of finger recognized per zone, as it informs about the noise introduced either by the hand's position, the arm being close to the table, a piece of clothes laying on the table... Figure 6.1 shows an example of data collected during the first part of the test.

Five fingers per area

This time, the test user is asked to place the five fingers of one hand on every areas. The main difference between placing one and five fingers comes from the noise introduced by the hand's palm. As it is closer to the table's surface, more infrared light is reflected, making the finger detection more difficult. Figure 6.2 illustrates the data collected with user 9.

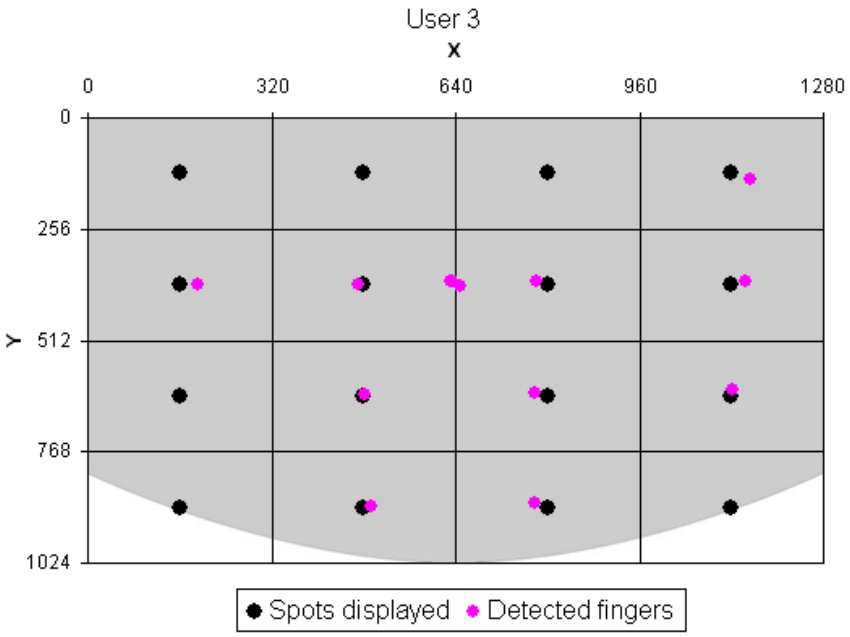


Figure 6.1: An example of data collected for the single finger test

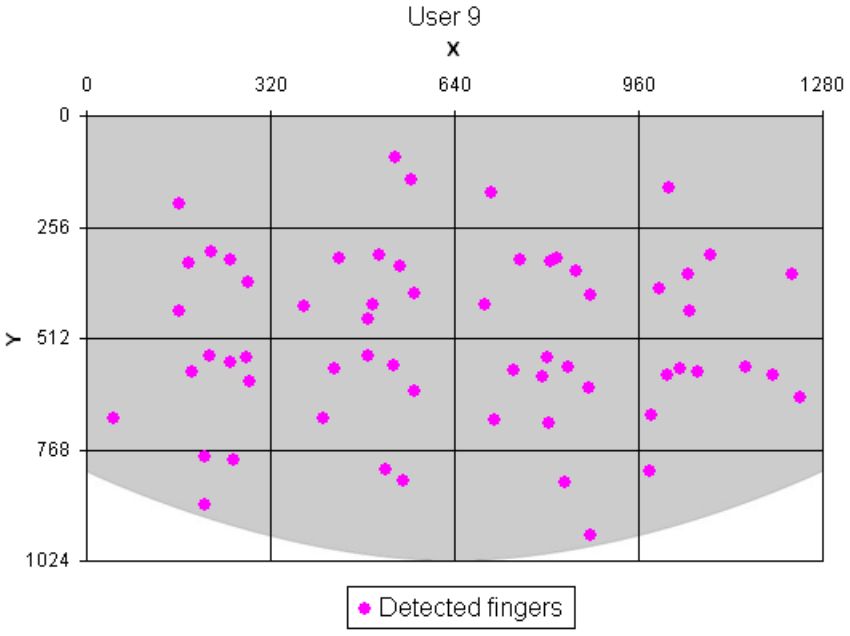


Figure 6.2: An example of data collected for the five fingers test

6.1.2 Interpreting the results

The results are interpreted as follows: in the first test, each area is given a note ranged from 0-2 depending on the number of fingers detected in the zone. 0 means that no finger have been detected, 2 means that the expected finger has been correctly detected, and 1 means that the finger has been detected but introduced noise resulting in the detection of an extra finger. For example in Figure 6.1, the two center areas on the second row (starting to the top) present an unexpected finger in the very center of the table's surface, probably due to the projector's reflection.

Concerning the second test, every area has been evaluated using the same 0-2 scale: 0 means that the recognition rate is very poor (less than four fingers detected out of the five expected), 2 means that the five fingers have been correctly detected, and 1 means that either one finger is missing, or the system considered noise as fingers, detecting more than five in the same area. Table 6.1 shows the percentage of correct recognition for each area for both tests. The results are presented according to the layout introduced in Figures 6.1 and 6.2. The percentages are computed as follows: for each area, the mean grade of all test users is divided by 2 and converted into a percentage value. It means that an area where all test users received the grade 0 result in a 0% recognition rate, while an area where all test users received a 2 results in a 100% recognition rate.

Single finger test			
9%	14%	23%	23%
82%	82%	73%	73%
73%	82%	92%	91%
36%	91%	86%	18%
Overall: 60% *Noise: 18%*			

Five fingers test			
0%	9%	41%	0%
64%	68%	64%	55%
59%	82%	86%	64%
23%	82%	64%	8%
Overall: 48% *Noise: 40%*			

Table 6.1: Finger recognition percentages for the two tests: the noise does not consider areas presenting poor detection

The first conclusion to be extracted from these results concerns the difference between the areas located in the center of the surface and the areas located in the corners or to the top of the surface. The results are much better in the central zones than to the more extreme ones. The low percentage of detection for the all top row can be explained by both the difficulty for the test user to reach the area keeping a comfortable position and the poor quality of this area in term of infrared light quantity and noise.

Comparing the two tests, the results are clearly better for the single finger version than for the five fingers one. This is due to the noise introduced by placing the whole hand close to the table. Indeed, considering only the areas with good finger detection rate, while the first test presents only 18% of areas implying noise, the second test's percentage of noise reaches 40%. This means that the user will have to adapt his manner of using the table to optimize the finger recognition rate.

Combining the two test results sets, the table's surface can be split into four kind of zones depending on the finger recognition rate it presents. Figure 6.3 illustrates the presence of high finger recognition rate in the lower-center part of the surface, which corresponds to the application's active area.

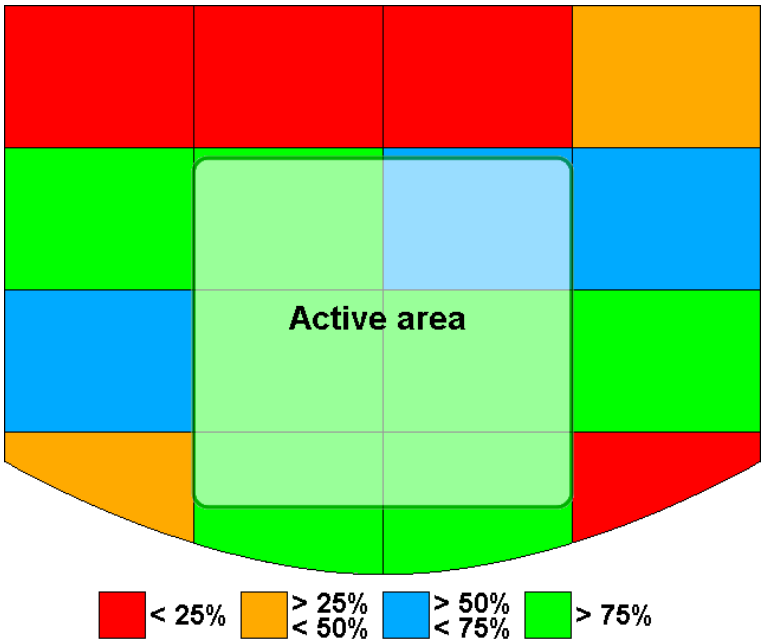


Figure 6.3: Finger detection test's final result: the application's active area matches the surface's areas with high finger recognition rate

6.2 Music-color association's evaluation

6.2.1 Test presentation

This test aims to understand someone's reaction when asked to associate music with colors. During the test, eleven persons with a technical background in computer science aged from 20-35 are asked to answer twelve questions classified in three categories (four questions per category). For each question, only one proposition is correct regarding to demeTouch. No information are provided prior to the test in order to collect intuitive results. For the same reason, the test user cannot go back and modify an answer he has already given.

From a musical genre to a color

The four first questions consist in choosing the color that best match the user representation of a musical genre in term of color. At this time only the hue is tested, the dynamic and energy being set to 100% each. In the example provided in Figure 6.4, the following colors are proposed: (orange), (green), (blue) and (pink).

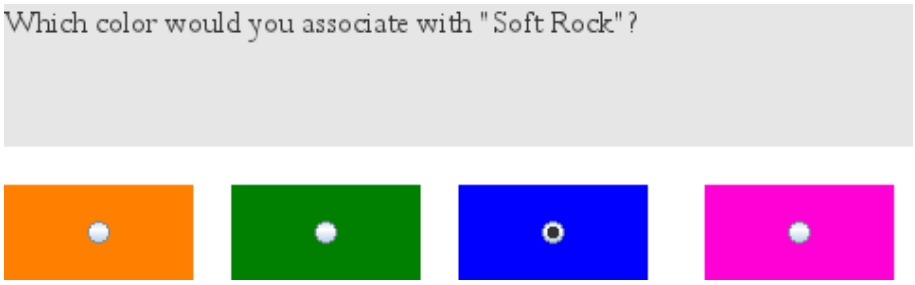


Figure 6.4: Associating a musical genre with a color

From musical features to color features

This category focuses on the dynamic and energy levels. The four propositions offer the same color differing in saturation and value. For example Figure 6.5 presents the hue value 120 (green) taking successively the following saturation-value levels: (20%-20%), (20%-80%), (80%-80%) and (80%-20%).

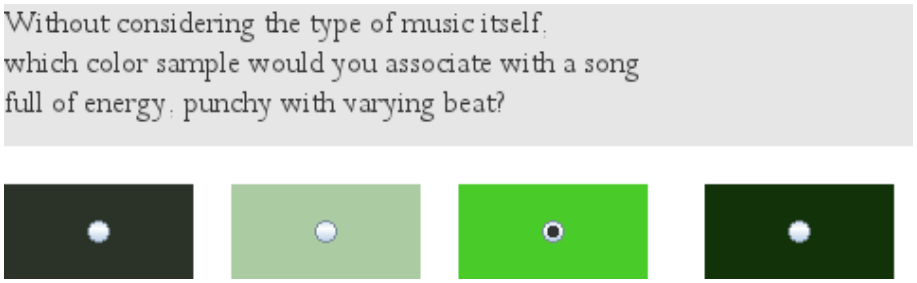


Figure 6.5: Associating musical features with color features

Mixing the parameters

The last four questions ask the user to picture himself into everyday life situations and to choose the color that in his opinion best match this situation. This time the three color features are mixed, as depicted in Figure 6.6, where the following colors are proposed: (*hue:* orange, *saturation:* 80%, *value:* 20%), (*hue:* green, *saturation:* 20%, *value:* 20%), (*hue:* pink, *saturation:* 20%, *value:* 80%) and (*hue:* cyan, *saturation:* 80%, *value:* 80%).

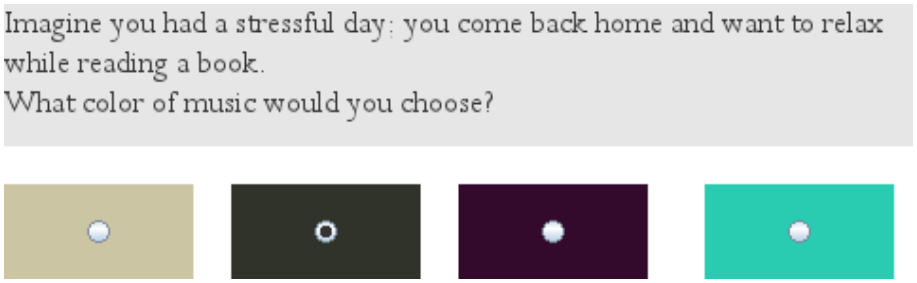


Figure 6.6: Associating a mood with a color

6.2.2 Interpreting the results

First of all, the group of persons who took part in this study cannot be considered absolutely representative as it is small (eleven persons) and composed of individuals with the same background. Nevertheless, it is sufficient to draw interesting conclusions on people's behavior. In addition, the results must be analyzed keeping in mind the high subjectivity of the questionnaire. Table 6.2 shows the results in percentage of correct answers for the all twelve questions, classified by category. The results given by the

tested persons are given a grade using the same 0-2 scale used for the finger detection test. While 0 corresponds to a wrong answer, 2 corresponds to a correct one, 1 being used to take into consideration the questionnaire's subjectivity. For example the correct answer for question 10 concerning the background for a conversation is "orange" (quiet classical music). While violet (Electro / techno) is considered as a wrong answer, "blue" (soft rock) is acceptable as it could also match such a situation depending on the user taste.

Questions			Correct answers		Results overall	Results detail
			color			
1	*Which color would you associate with*	Soft rock	blue		44%	45%
2		Classical	yellow			50%
3		Pop / soul	green			36%
4		Electro / techno	violet			45%
			*sat**	*val**		
5	*Which color would you associate with the musical features*	Punchy, varying beat	80%	80%	56%	82%
6		Calm, quiet	20%	20%		45%
7		Varying beat, few instruments	80%	20%		55%
8		Constant tempo, vivid	20%	80%		41%
			color *sat**	*val**		
9	*Imagine you would like to listen to music for*	Relaxing, reading	light green 20%	20%	55%	55% 36%
10		Discussing with guests	orange 20%	20%		45% 59%
11		Warming up before a party	violet 80%	80%		68% 82%
12		Doing some workout	pink 20%	80%		41% 55%

Table 6.2: Questionnaire results * *(sat = saturation, val = value)*

The first observation to be extracted from these results is that people intuitively answer more correctly to the questions related to the color features (second category of questions) than to the questions about the color itself (first category). However, when all parameters are mixed together (third category), the percentage of correct answers regarding the color's choice increases. This means that if people are not used to associate a music with a color, they intuitively associate more easily the concepts of dynamic and energy with saturation and value.

A closer look at the second category of questions illustrates the tendency of people to avoid dark and gray samples, associated with sad and unpleasant atmosphere. For example in question 7 most answers give the correct value of saturation describing the dynamic. However, at the same time, people tend to choose higher energy. This suggests that the most important feature is the varying rhythm associated with intense colors, neglecting the "few instruments". On the other hand, question 5 shows that when the correct answer includes high energy and high dynamic, the percentage of correct answers is close to 100% (considering the number of test users). This means that people easily choose vivid colors, associated with happy and cheerful mood.

The final conclusion that can be inferred from this test is that users should be introduced to the color-music association prior to using the application. In addition, the application itself should guide them (at least at the beginning) by displaying information about how to associate the color parameters to the music ones.

6.2.3 Additional test

To test if educating the user to the color-music association is useful, one of the tested persons is given some information about how the system works. Then, considering that he knows to which music style correspond the colors and how dynamic and energy are related to saturation and value, this person is asked to answer the questionnaire. Table 6.3 shows the results, illustrating the utility of this knowledge.

Questions	% correctness	Overall
1-4	100%	81%
5-8	88%	
9-12	69%	

Table 6.3: Results after education: the tested person gives much better results after being introducing to the music-color association

CHAPTER 7

Conclusions and future work

This chapter first sums up the possibilities offered by the system developed. Then it confronts the aim presented in Introduction with the result achieved at the end of the project. The author comments these achievements in a rather objective way, based on external opinions, and provides ideas for future improvements. Finally, the main challenges encountered during the project are summed up.

7.1 Project achievements

7.1.1 Finally, what does demeTouch propose?

In its development stage when printing this report, demeTouch offers a finger controlled music player, which relies on a color-based representation of the user's audio library. In other words, the user can go through the use cases described below. In those configurations, she uses her fingers to interact with the system by "clicking" on the table's surface, while she is standing in front of it.

- She creates a color-based playlist by picking selecting a sequence of colors and adjusting two color features that correspond to music characteristics. By doing so she defines the profile of the playlist she is about to listen to (for example from quiet and calm classical music to punchy and full of energy pop music).
- She consults her whole media library, ranged according either to one of the color features or one of the editorial data. She selects some of these pieces of music and starts playing the selection.
- While playing the desired sequence of music, she can navigate forward and backward through the playlist, change the playing position in the current track, pause and resume it, adjust the volume.

7.1.2 Conclusion and possible improvements

The aim of the project presented in Introduction was to combine a tabletop display with a finger recognition engine to manipulate a music library based on a color-based representation. If the general goal aimed has been reached, each of the three main component forming the overall system could be improved as follows.

Upgrading the table

In its current configuration, the tabletop display lacks from a certain instability. Inside the table structure, the components should be fixed together to avoid going through the calibration process every time the table slightly moves, and to allow the whole setup to be moved as a whole.

In addition, the interior of the wooden structure would benefit from being painted in black, to decrease the level of light reflections captured by the camera. For the same purpose, the glass plate should undergo an anti-reflection treatment.

Increasing the finger recognition rate

If the previous modifications themselves would definitely increase the image processing engine performances, new algorithms should be added to the finger detector to enhance its reliability. For example it could consist in controlling the detected spots' shape to discard the non finger-like ones, or tracking the fingers movements to avoid sporadic noise.

As stated in chapter 3 when justifying the choice of a tabletop display instead of a touch screen, the next step for demeTouch's image processing engine would be to implement a tangible recognition engine, that would allow the user to combine the use of fingers with objects. Depending on their nature and location, they could adjust the audio stream properties, navigate through the playlist or the library, or instantly switch to a specific color.

Enriching demeTouch's graphical user interface

The music application offers a wide space for improvements, depending on the user needs and the developer imagination. Before implementing additional features, the whole application would benefit from being intensively tested and debugged, as it sometimes adopt an unexpected behavior. In addition the concept of separating display area and active area could be explored to make the user experience clearer, easier and nicer.

The current finger detector triggers the event corresponding to the location extracted every time a finger is detected. Three improvements could be implemented in the applicative layer to enhance the system: first, a tracking algorithm could be developed to allow a smoother interaction with the interface. The user could "draw" the playlist profile, some key movements could be associated with predefined common actions such as switching panels, pausing the current track... Secondly, fingers could reproduce a mouse behavior, by implementing both the *press* and *release* actions to add to the current *click*. Finally, a real multi-touch solution would allow the user to select a range of colors from the playlist creator, or a range of tracks in the library.

Concerning the general design, different "skins" could be created to let the user decide on the interface general look. Concerning the player itself, the equalizer should be implemented in priority to respect the actual display. Then the following extra features could be considered: fading the played track in and out would smooth the link with the previous and following ones. Considering the use of tangibles in addition to fingers, the

player could include a disc-jockey environment that would allow the user to mix several pieces of music, adjusting the audio feedback in details, map this audio feedback with graphical effects transforming demeTouch in a real VJ (Visual Disc Jockey) platform...

7.1.3 Main technical issues

Using a tabletop display as a support for numerical data manipulation requires to handle substantial issues due to the nature of the device. Dealing with such a challenging device requires paying more attention to details than with standard output devices (such as screens) to make it function. In particular, the table instability makes the use of image processing more challenging in term of noise removal and calibration. Developing and optimizing the finger recognition engine implied to respect a step by step approach, as any change in an algorithm previously developed resulted in a consequent modification of the following ones. As a consequence, this project's most challenging issue was to combine and optimize simple algorithms to create a finger detector robust enough to deal the tabletop display.

Concerning the development of the music application, the main difficulty was to make the finger detector and the application run in parallel while communicating together, which introduced multi-threading issues. In addition, the application required to communicate with technologies such as XML or MySQL from the Java™ code. It implied using specific libraries dedicated to each technology.

7.1.4 Project management issues

The unusual format of this project, starting with a collaboration and finishing in a more personal work was desired by the author. If collaboration provides a certain feeling of security, working alone is more challenging as it requires more self-discipline. The author wants to emphasize his wish prior to the project to test this ability. Finally, the period of collaboration balanced the period of personal work, offering the opportunity to experience and compare both working environments.

Bibliography

[AR05] Samina T. Yousuf Azeemi and S. Mohsin Raza. A critical analysis of chromotherapy and its scientific evolution. *Evidence-based complementary and alternative medicine : eCAM*, 2(4):481–488, 2005.

[Bou07] Frédéric Boulet. Pianotable: Musical interaction on a tabletop display. Master's thesis, Aalborg University, 2007.

[CIS+06] Sheelagh Carpendale, Tobias Isenberg, Stacey D. Scott, Uta Hinrichs, André Miede, Russel Kruger, Stefan Habelski, and Kori Inkpen. Collaborative interaction on large tabletop displays. In *Conference Supplement of CSCW 2006: ACM Conference on Computer-Supported Cooperative Work*, pages 57–58, Banff (Alberta), Canada, 2006.

[CP95] Jean-Pierre Cocquerez and Sylvie Philipp. *Analyse d'images : filtrage et segmentation*. Masson, 1995.

[Fra91] Maarten Franssen. The ocular harpsichord of louis-bertrand castel: The science and aesthetics of an eighteenth-century *Cause Célèbre*. *Tractrix: Yearbook for the History of Science, Medicine, Technology and Mathematics*, pages 15–77, 1991.

[Inc99] Sun Microsystem Inc. *Programming in Java™ Advanced Imaging*, 1999.

[JFG06] Guillaume Jacquemin, Alexandre Fleury, and Vincent Gilbert. Demelo. STUDENTS PROJECT, realized at the Ecole Centrale d'Electronique, Paris, France, 2006.

[JKR06] Jesper Graarup Jensen, Simon Kofod, and Fabio Righi. 3d interaction on a tabletop display. Technical report, Aalborg University, 2006.

[KB07] Martin Kaltenbrunner and Ross Bencina. reactivision: A computer-vision framework for table-based tangible interaction. In *1st International Conference on Tangible and Embedded Interaction (TEI07)*, Baton Rouge (Louisiana) USA, 2007.

[KJGA06] Martin Kaltenbrunner, Sergi Jordà, Günter Geiger, and Marcos Alonso. The reactable*: A collaborative musical instrument. In *15th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'06)*, pages 406–411, Manchester, U.K, 2006.

[Lar05] Søren Larsen. Puka finder: an introduction to tabletop displays. Master's thesis, Aalborg University, 2005.

[SGVF05] Ian Stavness, Jennifer Gluck, Leah Vilhan, and Sidney Fels. The musictable: A map-based ubiquitous system for social interaction with a digital music collection. In *4th International Conference on Entertainment Computing (ICEC2005)*, pages 291–302, Sanda, Japan, 2005.

[Wal91] Morton Walker. *The Power of Color*. Avery Publishing Group, 1991.

List of Figures

List of Tables

APPENDIX A

Colophon

This report has been written in LaTeX using the MiKTeX 2.5 distribution under TeXnicCenter 1 Beta 7.01. The layout of the report has been designed by its author with the help of Flemming Kjær Jensen's report template. It reuses the class file `iesreport.cls` provided by the Aalborg University's Department of Communication. A copy of the template can be downloaded from `http://kom.aau.dk/net/courses/latex/`. The author of this report has modified this original template to match his own requirements. The layout style of this report is a result of discussions with and feedback from students at the Department of Electronic Systems. The cover of a master's thesis report at Aalborg University often reflects the product that is created during the project period, and the front page images on the cover should not be reused.

Images have been created using GIMP 2.2. Diagrams have been designed with Microsoft Visio© 2003. The Java code described in the report has been originally created with Eclipse SDK 3.2.0. The graphs, including trend line equations have been computed using Microsoft Excel© 2003.

Attached to the report, the reader will find a CD containing an electronic version of this document, the code source of the project, a demonstration video of demeTouch, and some photos of the physical setup. The reader should refer to the included *Readme* file for the details concerning the use of the CD content.

APPENDIX B

demelo in details

This appendix details the functionalities of demelo's application, demeTouch's predecessor. First the music player is presented, then the playlist creator with the color selector, and finally the library manager.

Music player

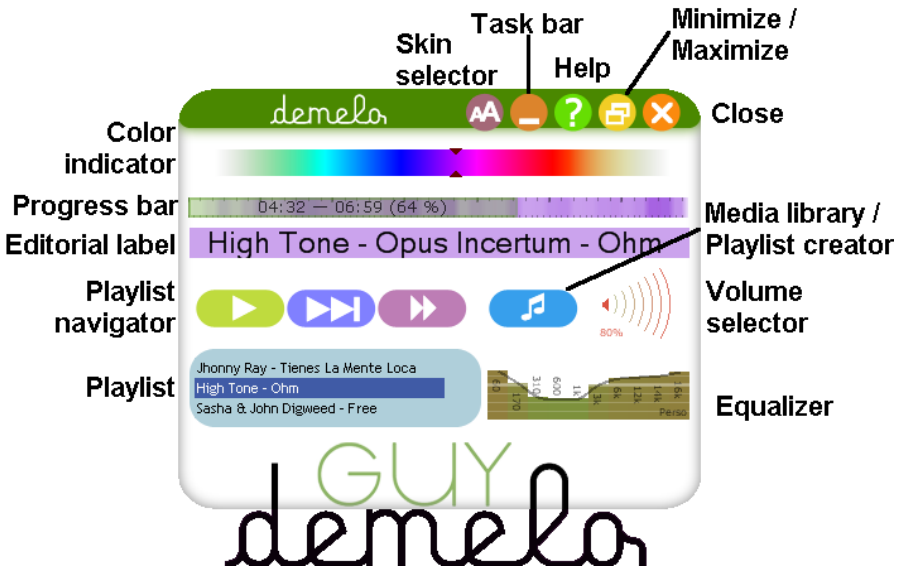


Figure B.1: demelo's music player in details

Top banner

This banner contains four buttons corresponding to basic functions related to the window's manipulation: minimizing the player in the task bar, maximizing it, closing the player, getting some help about the player. In addition, a fifth button allows the user to switch between three different skins.

Managing the playlist and manipulating the music

The playlist panel displays three tracks at a time. This choice is due to the concept of the player: the user is expected to think about music in term of sequence of colors matching a mood, an atmosphere, instead of manually switching from songs to others. By showing a limited part of the playlist, the user is less tempted to switch from a color to another totally different, which would put away the importance of the playlist creator. Above the playlist panel, three buttons allows the user to pause/resume the current track, go forward in the playlist by playing the next track, navigate further in the current track by changing the position in the piece of music. Changing the playing position in the current track can also be done by clicking on the desired position in the progress bar.

Two options are offered to the user to manipulate the audio stream: she can either set the volume from 0-100%, and use the equalizer to adjust a set of ten frequencies (from $60Hz$-$16kHz$). By double-cliking on the equalizer, the values are reset to the default ones.

Information about the current track

The color of the progress bar informs about the track's dynamic. For example a classical pop song has the following profile: darker zones stand for the chorus, while brighter zones represent the verses. Above the progress bar, the color indicator informs about the current track's color. Below the progress bar, the editorial panel informs about the track's title, artist, and album.

Playlist creator

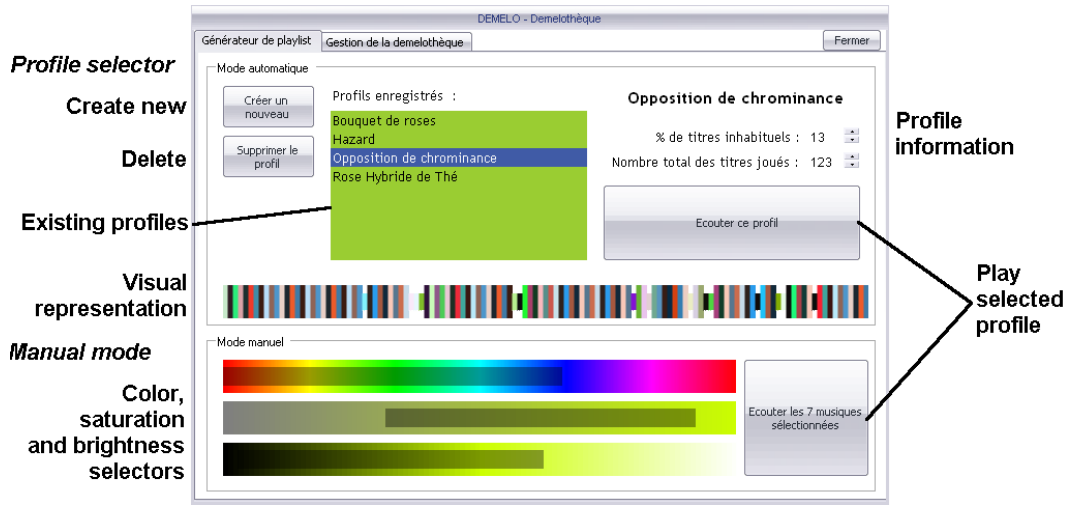


Figure B.2: demelo's playlist creator in details

demelo's playlist creator proposes two modes to create a playlist: the user can choose a profile from a set of predefined ones, or manually select the color features.

Profiles selector

In this panel a list of existing profiles is displayed. Once clicked on one of them, some information is displayed on the right-side of the window, informing about the percentage of tracks that does not exactly match the required colors (it can be seen as a “surprise” factor) and the number of tracks forming the playlist. These two parameters can be adjusted manually. In addition, the user can create her own profile, and delete one of the existing ones. The visual representaion illustrates the profile sequence of colors that will be played in order from left to right.

Manual mode

From this panel the user can manually choose the color parameters that will compose the playlist. She can independently select a range of colors, of saturations and brightnesses by pressing the mouse on the corresponding bar and moving up (to increase the selection) or down (to decrease the selection). The number of music pieces that match the selection is displayed on the right button. Once clicked, this button starts playing the created playlist.

Library manager

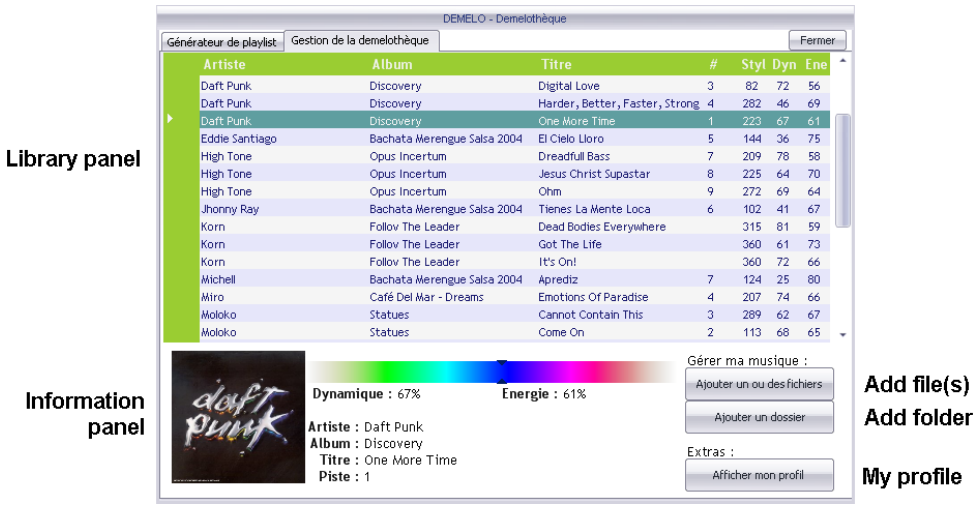


Figure B.3: demelo’s library manager in details

This panel lists all tracks known by the system and taken into consideration when creating a playlist. From this interface the user can add files or folders to the library. Every time a new file is added, its color features are extracted. In addition, a temporal and spectral prints of the song are processed, allowing to detect if the song is already part of the library (for example in a different folder with another name). To remove a track from the library, the user should select the corresponding line in the library panel and press the "delete" key on her keyboard.

The areas located below the library panel displays information about any title selected in the library panel. The CD cover (if available on local computer), the three color parameters, and some editorial information: artist, title, album, track number.

The "My profile" button on the bottom right corner allows the user to get a visual representation of her media library, split into bars of color which height depends on the number of title matching this color. An example is provided in Figure B.4.

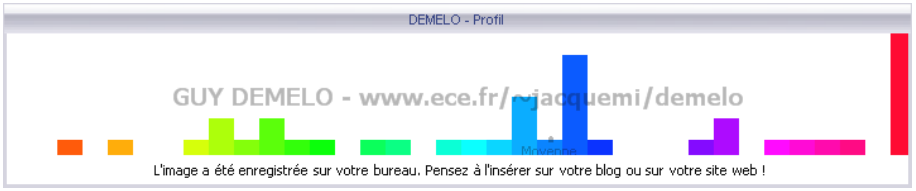


Figure B.4: Visual representation of the user library